Basic Theory

OeGOR Summer-School 2024, Krems

Timo Gschwind
Lehrstuhl für BWL, insb. Logistik

Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

July 22 – 26, 2024

Agenda

1. Column Generation
2. Dantzig-Wolfe Reformulation
3. Branching and Cutting

PART I

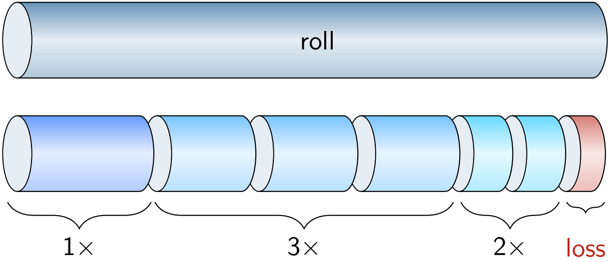
Column Generation

=

An algorithm to solve LPs with many(!) variables (=columns).

Cutting Stock Problem:

- L length of rolls to cut
- m pieces with shorter lengths $\ell_1, \ldots, \ell_m \leq L$
- $b_1, \ldots, b_m$ demands for pieces
- Goal: Minimize number of rolls

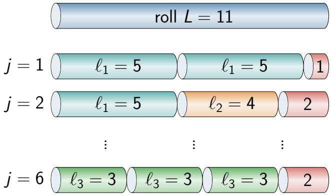


Cutting Stock

Example: Roll $L = 11$, pieces $\ell = (\ell_1, \ell_2, \ell_3)^\top = (5, 4, 3)^\top$

Cutting patterns (efficient, i.e., loss < 3):

Pattern j	$\ell_1 = 5$ #	$\ell_2 = 4$ #	$\ell_3 = 3$ #	Loss r_j
1	2	0	0	1
2	1	1	0	2
3	1	0	2	0
4	0	2	1	0
5	0	1	2	1
6	0	0	3	2



Here: Only $n = 6$ efficient patterns.

$$A_1 = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, A_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, A_3 = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}, A_4 = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}, A_5 = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}, A_6 = \begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix}$$

Example (cont'd): Demand $b = (b_1, b_2, b_3)^\top = (100, 400, 300)^\top$.

Model:

$$
\begin{array}{lrrrrrrl}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & \\
\text{s.t.} & 2\lambda_1 & +1\lambda_2 & +1\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 100 \\
 & 0\lambda_1 & +1\lambda_2 & +0\lambda_3 & +2\lambda_4 & +1\lambda_5 & +0\lambda_6 & \geq 400 \\
 & 0\lambda_1 & +0\lambda_2 & +2\lambda_3 & +1\lambda_4 & +2\lambda_5 & +3\lambda_6 & \geq 300 \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6 & \geq 0
\end{array}
$$

Modell, Gilmore & Gomory (1961, 1963): $A = (a_{ij}) = (A_1, \ldots, A_n)$

$$
z = \min \sum_{j=1}^{n} \lambda_j \tag{1}
$$

$$
\text{s.t.} \quad \sum_{j=1}^{n} a_{ij}\lambda_j \geq b_i \quad i = 1, 2, \ldots, m \tag{2}
$$

$$
\lambda_j \geq 0 \quad j = 1, 2, \ldots, n \tag{3}
$$

Remark: This is an LP! **Later:** integer problems: λ_j integer...

Example: Roll L to be cut into pieces
$\ell = (\ell_1, \ell_2, \ell_3, \ell_4, \ell_5)^\top = (10, 8, 5, 4, 3)^\top$

Number of patterns n (*#patterns*) depending on L?

L	*#patterns*	L	*#patterns*	L	*#patterns*
10	8	12	10	14	14
16	19	18	25	20	33
25	57	30	94	35	143
40	216	35	307	50	429
60	770	70	1282	80	2015
90	3024	100	4371	110	6124
120	8358	130	11153	140	14596
150	18780	200	54516	250	126344
300	252936	350	456870	400	764631

Linear Program (LP) in standard form:

$$\begin{aligned} \min c^\top \lambda & \\ A\lambda &= b \\ \lambda &\geq \mathbf{0} \end{aligned}$$

If an LP has at least one optimal solution, then there also exists a feasible basic solution which is an optimal solution.

Main idea of Simplex algorithm (G. Dantzig, 1947):

1. Start in feasible basic solution
2. Iteration: Go to a better feasible basic solution, if existing

Definition (basis, basic variables, feasible basic solution)

A **basis** B is an invertible submatrix of A. We can reorder the columns of A such that $A = (B|N)$. Let $\lambda = (\lambda_B|\lambda_N)^\top$ and $c^\top = (c_B|c_N)$ be the corresponding partitions of x and c. The variables λ_B are **basic variables**, the variables λ_N are **non-basic variables**. A **feasible basic solution** is a feasible solution with $\lambda_N = \mathbf{0}$.

We can rewrite:

$$
\begin{aligned}
& A\lambda = (B|N)\begin{pmatrix}\lambda_B \\ \lambda_N\end{pmatrix} = b \\
\Longrightarrow \quad & B\lambda_B + N\lambda_N = b \\
\Longrightarrow \quad & \lambda_B + B^{-1}N\lambda_N = B^{-1}b \\
\Longrightarrow \quad & \boxed{\lambda_B = B^{-1}b - B^{-1}N\lambda_N}
\end{aligned}
$$

$$c^\top\lambda = c_B^\top\lambda_B + c_N^\top\lambda_N$$

Impact of basis change on cost:

$$\begin{aligned} c^\top \lambda &= c_B^\top \lambda_B + c_N^\top \lambda_N \\ &= c_B^\top \left(B^{-1} b - B^{-1} N \lambda_N \right) + c_N^\top \lambda_N \\ &= \underbrace{c_B^\top B^{-1}}_{\substack{= \pi^\top \\ (\textit{dual sol.})}} b + \underbrace{(c_N^\top - c_B^\top B^{-1} N)\lambda_N}_{\substack{\text{components:} \\ (c_j - c_B^\top B^{-1} A_j)\lambda_j}} \end{aligned}$$

Definition (reduced cost)

The **reduced cost** of a variable λ_j is $\tilde{c}_j(\pi) = c_j - c_B^\top B^{-1} A_j = c_j - \pi^\top A_j$.

- If $\tilde{c}_j(\pi) < 0$ for some $\lambda_{j^\star}$
  $\rightarrow$ (one of those) $\lambda_j^\star$ should enter the basis
  $\rightarrow$ Dantzig's criterion: select $\lambda_{j^\star}$ such that $j^\star = \arg\min_j \tilde{c}_j(\pi)$
- If $\tilde{c}_j(\pi) \geq 0$ for all λ_j
  $\rightarrow$ current solution is optimal!
- Simplex explicitly computes $\tilde{c}_j(\pi)$ for all λ_j in each iteration

Column Generation – Principle

Instead of all variables λ_j, $j \in J$ ...

$$
\begin{array}{lrcl}
 & z_{MP} = \min c_J^\top \lambda_J & & \\
(MP) & A_J \lambda_J & = & b \\
 & \lambda_J & \geq & 0
\end{array}
$$

"Master Program (MP)"

... use subset of variables λ_j, for $j \in J' \subset J$

$$
\begin{array}{lrcll}
 & z_{RMP} = \min c_{J'}^\top \lambda_{J'} & & & \\
(RMP) & A_{J'} \lambda_{J'} & = & b & [\pi] \\
 & \lambda_{J'} & \geq & 0 &
\end{array}
$$

"Restricted Master Program (RMP)"

Does there exist a variable $\lambda_{j^\star}$ for $j^\star \in J \setminus J'$ with negative reduced cost (rdc)? This is the *"pricing problem (PP)"*:

$$
(PP) \qquad \tilde{c}^\star(\pi) = \min_{j \in J} \left(c_j - \pi^\top A_j \right) \overset{?}{<} 0
$$

yes: Add $j^\star$ to J' (*add column* $A_{j^\star}$) and re-optimize RMP

no: Done! MP solved!

Given: Dual solution $\pi = (\pi_1, \pi_2, \ldots, \pi_m)^\top \geq 0$ of RMP

Reduced cost of pattern $A_j = (a_{ij})_{i=1,\ldots,m}$:

$$\tilde{c}_j(\pi) = 1 - \sum_{i=1}^{m} \pi_i a_{ij}$$

Task: Find a pattern A_j with minimum $\tilde{c}_j(\pi)$!

$\rightarrow$ explicit enumeration of all patterns with their rdc is not practical

$\rightarrow$ instead: implicit enumeration by solving auxiliary optimization problem

Column Generation for Cutting Stock – Pricing Problem

Feasible Patterns: A pattern $A_j = (a_{ij})_{i=1,\dots,m} \in \mathbb{Z}_+^m$ is feasible if

$$\sum_{i=1}^{m} \ell_i a_{ij} \leq L.$$

Substituting a_{ij} (index j is irrelevant) by new variables x_i, $i = 1, 2, \dots, m$:

Pricing Problem = Subproblem

$$\begin{aligned} \tilde{c}^\star(\pi) \quad = \quad & \min \quad 1 - \sum_{i=1}^{m} \pi_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} \ell_i x_i \leq L \\ & x_1, \dots, x_m \in \mathbb{Z}_+ \end{aligned}$$

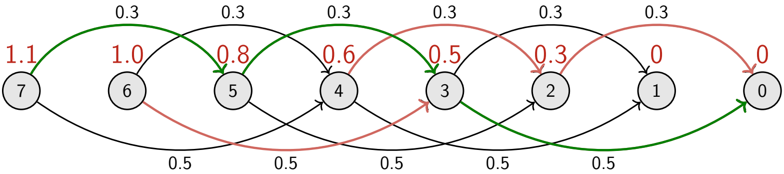
Integer Knapsack Problem

$$\begin{aligned} z_{IKP}(\pi) \quad = \quad & \max \sum_{i=1}^{m} \pi_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} \ell_i x_i \leq L \\ & x_1, \dots, x_m \in \mathbb{Z}_+ \end{aligned}$$

Excursus (solution of integer knapsack problem):

Example: Roll $L = 7$, $m = 2$ pieces, $\ell = (2, 3)^\top$,
dual solution $\pi = (\pi_1, \pi_2)^\top = (0.3, 0.5)^\top$

Solution by dynamic programming:



- Solution is pattern $(2, 1)$ and $z_{IKP} = 1.1$
- Reduced cost of this pattern is $\tilde{c}^\star(\pi) = 1 - 1.1 = -0.1 < 0$
- Linear relaxation of Master Program (MP) not yet solved. Iterate:
  - $\rightarrow$ re-optimize RMP with new pattern
  - $\rightarrow$ solve pricing problem with new dual prices again

Task: Provide feasible basic solution to RMP

$\rightarrow$ does not necessarily have to be feasible for *real problem*

Two common alternatives:

feasible initial solution	may be difficult in some applications
big-M	simple to implement!

Example: Roll $L = 102$, $m = 5$ pieces of length $\ell = (10, 8, 5, 4, 3)^\top$ with demand $b = (37, 920, 177, 422, 899)^\top$.

Initial feasible solution: one pattern for each piece $i \in \{1, 2, \ldots, m\}$

- take piece i with maximal multiplicity $\lfloor L/\ell_i \rfloor$
- fill remaining "space" with smallest piece so that loss $< \min_k \ell_k$

Initial patterns:

$$\begin{pmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 12 \\ 0 \\ 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 20 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 25 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 34 \end{pmatrix}$$

Restricted master program (RMP):

$$
\begin{array}{llllllll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941			

Restricted master program (RMP):

$$\begin{array}{lrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \geq 0 &
\end{array}$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = A_6$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & \geq 899 & [\pi_5] \\
& \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = A_6$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941			

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrlr}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1, 0, 0, 23, 0) = A_7$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$

Column Generation for Cutting Stock – Example

Restricted master program (RMP):

$$
\begin{array}{lccccccccl}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941			

Restricted master program (RMP):

$$
\begin{array}{llllllll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0, 0, 18, 3, 0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1, 0, 0, 23, 0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9, 0, 1, 1, 1) = A_8$

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = A_8$

Restricted master program (RMP):

$$
\begin{array}{llllllllllll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq & 37 & [\pi_1] \\
 & 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq & 920 & [\pi_2] \\
 & 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq & 177 & [\pi_3] \\
 & 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq & 422 & [\pi_4] \\
 & 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq & 899 & [\pi_5] \\
 & \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq & 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = A_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941			

Restricted master program (RMP):

$$
\begin{array}{llllllllll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = A_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941	1.0	0.0	

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = A_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941	1.0	0.0	Optimality!

Column Generation for Cutting Stock – Example

Restricted master program (RMP):

$$
\begin{array}{lrrrrrrrrll}
\min & 1\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +1\lambda_5 & +1\lambda_6 & +1\lambda_7 & +1\lambda_8 & & \\
\text{s.t.} & 10\lambda_1 & +0\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +1\lambda_7 & +9\lambda_8 & \geq 37 & [\pi_1] \\
& 0\lambda_1 & +12\lambda_2 & +0\lambda_3 & +0\lambda_4 & +0\lambda_5 & +0\lambda_6 & +0\lambda_7 & +0\lambda_8 & \geq 920 & [\pi_2] \\
& 0\lambda_1 & +0\lambda_2 & +20\lambda_3 & +0\lambda_4 & +0\lambda_5 & +18\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 177 & [\pi_3] \\
& 0\lambda_1 & +0\lambda_2 & +0\lambda_3 & +25\lambda_4 & +0\lambda_5 & +3\lambda_6 & +23\lambda_7 & +1\lambda_8 & \geq 422 & [\pi_4] \\
& 0\lambda_1 & +2\lambda_2 & +0\lambda_3 & +0\lambda_4 & +34\lambda_5 & +0\lambda_6 & +0\lambda_7 & +1\lambda_8 & \geq 899 & [\pi_5] \\
& \lambda_1, & \lambda_2, & \lambda_3, & \lambda_4, & \lambda_5, & \lambda_6, & \lambda_7, & \lambda_8 & \geq 0 &
\end{array}
$$

z_{RMP}	π_1	π_2	π_3	π_4	π_5	$z_{IKP}(\pi)$	$\tilde{c}^\star(\pi)$	pattern
128.028	0.1	0.07843	0.05	0.04	0.02941	1.02	-0.02	$(0,0,18,3,0) = A_6$
127.8314	0.1	0.07843	0.04889	0.04	0.02941	1.02	-0.02	$(1,0,0,23,0) = A_7$
127.4901	0.1	0.07843	0.04903	0.03913	0.02941	1.0176	-0.0176	$(9,0,1,1,1) = A_8$
127.451	0.09804	0.07843	0.04902	0.03922	0.02941	1.0	0.0	Optimality!

Try the same with big-M!

Column Generation for Cutting Stock – Example

Example (cont'd): The last RMP terminates with:

- Optimal (LP) objective value $z_{RMP} = 127.451$
- Use pattern $(0, 12, 0, 0, 2)$ exactly $\lambda_2 = 76.6667$ times 77
  $(0, 0, 0, 0, 34)$ exactly $\lambda_5 = 21.866$ times 22
  $(0, 0, 18, 3, 0)$ exactly $\lambda_6 = 9.7098$ times 10
  $(1, 0, 0, 23, 0)$ exactly $\lambda_7 = 16.9856$ times 17
  $(9, 0, 1, 1, 1)$ exactly $\lambda_8 = 2.2239$ times 3
- Sum $\Sigma = 129$

Implications for Integer Cutting Stock Problem (CSP):

- $\lceil 127.451 \rceil = 128$ is a lower bound for the integer CSP
- Rounding up the λ-values gives a feasible integer solution with value 129
- An optimal integer solution uses either 128 or 129 rolls

More on integer formulations and solutions in Part II

Some Takeaways

- Column generation allows to solve HUGE models (many variables)
  - $\rightarrow$ only a tiny fraction of the variables actually considered
  - $\rightarrow$ example: 3 variables generated (8 in total) of more than 4300
- There is a price to pay:
  - $\rightarrow$ iterate between re-optimization of RMP and PP
- PP is an optimization problem itself
  - $\rightarrow$ need to be solved often
  - $\rightarrow$ better have a good algorithm for that

PART II

Dantzig-Wolfe Reformulation

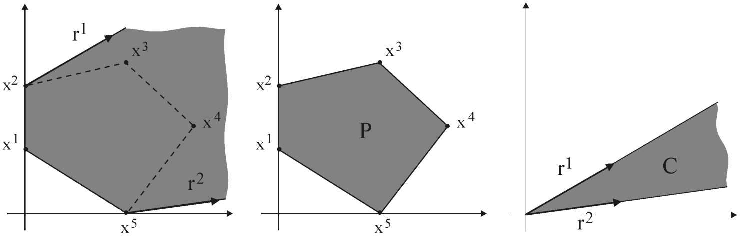
=

A systematic way to produce
extensive formulations (many columns)
from compact formulations.

Minkowski-Weyl Theorem (1897/1935):

For $X \subseteq \mathbb{R}^d$ the following statements are equivalent:

1. X **is a polyhedron**, i.e., for some matrix D and vector d, $X = \{x : Dx \leq d\}$;
2. There are finite vectors $\{x_p, p \in P\}$ and $\{x_r, r \in R\}$ in $\mathbb{R}^d$ such that $X = \text{conv}(x_p, p \in P) + \text{nonneg}(x_r, r \in R)$;
3. X is the sum of a polytope and a convex cone.



Polyhedra

Let $X = \{x : Dx \leq d\}$. Every $x \in X$ can be written as:

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r$$

with

$$\lambda_p \geq 0, p \in P \qquad \sum_{p \in P} \lambda_p = 1 \qquad \text{and} \qquad \lambda_r \geq 0, r \in R$$

For a polyhedron $X = \{Dx \leq d, x \geq \mathbf{0}\}$,

- $\{x_p, p \in P\}$ can be chosen as the set of extreme points of X
- $\{x_r, r \in R\}$ can be chosen as the set of extreme rays of X

Overview

Three types of Dantzig-Wolfe reformulation:

1. for Linear Programs (LP)
2. for (Mixed) Integer Programms (IP) by Convexification
3. for (Mixed) Integer Programms (IP) by Discretization

Dantzig-Wolfe Reformulation for LP

"Original model/formulation":

$$\begin{aligned} z_{LP} &= \min c^\top x \\ Ax &= b \\ Dx &= d \\ x &\geq \mathbf{0} \end{aligned}$$

$X = \{x : Dx = d, x \geq \mathbf{0}\}$

$\{x_p, p \in P\}$ extreme points of X

$\{x_r, r \in R\}$ extreme rays of X

Substitute

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r$$

Equivalent LP (master program): ***"extensive formulation"***

$$z_{MP} = \min \sum_{p \in P} (c^\top x_p)\lambda_p + \sum_{r \in R} (c^\top x_r)\lambda_r$$

$$\sum_{p \in P} (Ax_p)\lambda_p + \sum_{r \in R} (Ax_r)\lambda_r = b \qquad [\pi]$$

$$\sum_{p \in P} \lambda_p = 1 \qquad [\mu]$$

$$\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R$$

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r \qquad \leftarrow \boxed{\text{No need to keep in LP}}$$

D-W Reformulation for IP by Convexification

“Original model/formulation”:

$$\begin{aligned} z_{IP} &= \min c^\top x \\ Ax &= b \\ Dx &= d \\ x &\geq \mathbf{0}, \ \text{integer} \end{aligned}$$

$X = \{x : Dx = d, x \geq \mathbf{0} \ \text{ integer}\}$

$\{x_p, p \in P\}$ extreme points of $conv(X)$

$\{x_r, r \in R\}$ extreme rays of $conv(X)$

Substitute

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r$$

Equivalent IP (integer master program): ***“extensive formulation”***

$$z_{IMP} = \min \sum_{p \in P} (c^\top x_p)\lambda_p + \sum_{r \in R} (c^\top x_r)\lambda_r$$

$$\begin{aligned} \sum_{p \in P} (Ax_p)\lambda_p + \sum_{r \in R} (Ax_r)\lambda_r &= b \qquad [\pi] \\ \sum_{p \in P} \lambda_p &= 1 \qquad [\mu] \end{aligned}$$

$$\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R$$

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r \quad \text{integer} \ \leftarrow \ \boxed{\text{Important!}}$$

D-W Reformulation for IP by Discretization

"Original model/formulation":

$$\begin{aligned} z_{IP} &= \min c^\top x \\ Ax &= b \\ Dx &= d \\ x &\geq \mathbf{0}, \quad \text{integer} \end{aligned}$$

$X = \{x : Dx = d, x \geq \mathbf{0} \ \text{ integer}\}$

$\{x_p, p \in P\}$ integer points of X

$\{x_r, r \in R\}$ integer rays of X

Substitute

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r \quad \text{binary/non-neg. integer combinations}$$

Equivalent IP (integer master program): *"extensive formulation"*

$$z_{IMP} = \min \sum_{p \in P} (c^\top x_p)\lambda_p + \sum_{r \in R} (c^\top x_r)\lambda_r$$

$$\begin{aligned} \sum_{p \in P} (Ax_p)\lambda_p + \sum_{r \in R} (Ax_r)\lambda_r &= b \qquad [\pi] \\ \sum_{p \in P} \lambda_p &= 1 \qquad [\mu] \end{aligned}$$

$$\lambda_p \in \{0, 1\}, p \in P, \quad \lambda_r \in \mathbb{Z}_+, r \in R$$

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r \qquad \leftarrow \boxed{\text{redundant}}$$

Overview

Three types of Dantzig-Wolfe reformulation:

1. for Linear Programs (LP)
2. for (Mixed) Integer Programms (IP) by Convexification
3. for (Mixed) Integer Programms (IP) by Discretization

Dantzig-Wolfe Reformulation for LP

"Original model/formulation":

$$
\begin{aligned}
z_{LP} &= \min c^\top x \\
Ax &= b \\
Dx &= d \\
x &\geq \mathbf{0}
\end{aligned}
$$

$X = \{x : Dx = d, x \geq \mathbf{0}\}$

$\{x_p, p \in P\}$ extreme points of X

$\{x_r, r \in R\}$ extreme rays of X

Substitute

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r$$

Equivalent LP (master program): ***"extensive formulation"***

$$
\begin{aligned}
z_{MP} = \min \sum_{p \in P} (c^\top x_p)\lambda_p + \sum_{r \in R} (c^\top x_r)\lambda_r & & & \\
\sum_{p \in P} (Ax_p)\lambda_p + \sum_{r \in R} (Ax_r)\lambda_r &= b & & [\pi] \\
\sum_{p \in P} \lambda_p &= 1 & & [\mu] \\
\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R & & & \\
x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r & & &
\end{aligned}
$$

$\leftarrow$ No need to keep in LP

Dantzig-Wolfe Reformulation for LP

- Original and extensive formulations are equivalent!

$$z_{LP} = z_{MP}$$

- Extensive formulation (=MP) has huge number of variables $|P|+|R|$

Solution by column generation:

→ RMP solved with LP-solver (primal, dual, barrier)

> dual solution π, μ

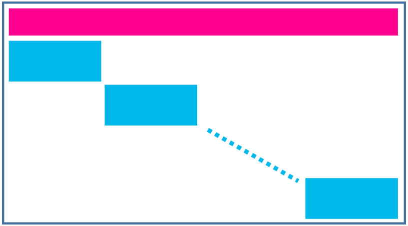
→ pricing (=sub)problem (PP):

$$\begin{aligned} \tilde{c}^\star(\pi,\mu) \quad &= \quad -\mu + \min(c^\top - \pi^\top A)x \\ \text{s.t.} \quad & \quad Dx = d \\ & \quad x \geq \mathbf{0} \end{aligned}$$

> extreme point(s) x_p or extreme ray(s) x_r with negative rdc
> partial pricing possible! (→ not necessary to solve PP to optimality)

- Allows exploiting that D has block-diagonal structure ...

Block-diagonal structure:



$$
\begin{array}{rcllllll}
z_{LP} & = & \min & c^{1\top}x^1 + c^{2\top}x^2 + \cdots + c^{\bar{k}\top}x^{\bar{k}} & & & & \\
& & & A^1x^1 & + \; A^2x^2 & + \cdots + & A^{\bar{k}}x^{\bar{k}} & = b \\
& & & D^1x^1 & & & & = d^1 \\
& & & & D^2x^2 & & & = d^2 \\
& & & & & \ddots & & \\
& & & & & & D^{\bar{k}}x^{\bar{k}} & = d^{\bar{k}} \\
& & & x^1, & x^2, & \ldots, & x^{\bar{k}} & \geq \mathbf{0}
\end{array}
$$

- $Dx = d$ decomposes into blocks

$$
D^1x^1 = d^1 \qquad D^2x^2 = d^2 \qquad \cdots \qquad D^{\bar{k}}x^{\bar{k}} = d^{\bar{k}}
$$

- Individual Dantzig-Wolfe reformulation for each block

  $\rightarrow$ new variables for each block

  $\rightarrow$ one subproblem per block

Dantzig-Wolfe Reformulation for LP

Let $K = \{1, 2, 3, \ldots, \bar{k}\}$. **Equivalent LP** for the block-diagonal case:

$$
\begin{array}{rcll}
z_{MP} = \min \displaystyle\sum_{k \in K} \sum_{p \in P^k} (c^{k\top} x_p^k) \lambda_p^k + \sum_{k \in K} \sum_{r \in R^k} (c^{k\top} x_r^k) \lambda_r^k & & & \\
\displaystyle\sum_{k \in K} \sum_{p \in P^k} (A^k x_p^k) \lambda_p^k + \sum_{k \in K} \sum_{r \in R^k} (A^k x_r^k) \lambda_r^k & = & b & [\pi] \\
\displaystyle\sum_{p \in P^k} \lambda_p^k & = & 1 \qquad k \in K & [\mu_k] \\
\lambda_p^k \geq 0, k \in K, p \in P^k, \qquad \lambda_r^k \geq 0, k \in K, r \in R^k & & & \\
x^k = \displaystyle\sum_{p \in P} \lambda_p^k x_p^k \qquad + \sum_{r \in R} \lambda_r^k x_r^k \qquad k \in K & & &
\end{array}
$$

One subproblem for each $k \in K$:

$$
\begin{array}{rcl}
\tilde{c}^{k\star}(\pi, \mu^k) = -\mu^k & + & \min (c^{k\top} - \pi^\top A^k) x^k \\
\text{s.t.} & & D^k x^k = d^k \\
& & x^k \geq \mathbf{0}
\end{array}
$$

If all blocks are identical: **Aggregate** $\rightarrow A^{\bullet}, x^{\bullet}, c^{\bullet}, D^{\bullet}, d^{\bullet}$

$$
\begin{aligned}
z_{MP} = \min \sum_{p \in P} (c^{\bullet\top} x_p^{\bullet}) \lambda_p + \sum_{r \in R} (c^{\bullet\top} x_r^{\bullet}) \lambda_r & & & \\
\sum_{p \in P} (A^{\bullet} x_p^{\bullet}) \lambda_p + \sum_{r \in R} (A^{\bullet} x_r^{\bullet}) \lambda_r & = & b & \quad [\pi] \\
\sum_{p \in P} \lambda_p & = & |K| & \quad [\mu] \\
\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R & & & \\
x^{\bullet} = \sum_{p \in P} \lambda_p x_p^{\bullet} + \sum_{r \in R} \lambda_r x_r^{\bullet} & & &
\end{aligned}
$$

One subproblem only:

$$
\begin{aligned}
\tilde{c}^{\bullet\star}(\pi, \mu) = -\mu \quad + \quad & \min (c^{\bullet\top} - \pi^{\top} A^{\bullet}) x^{\bullet} \\
\text{s.t.} \quad & D^{\bullet} x^{\bullet} = d^{\bullet} \\
& x^{\bullet} \geq \mathbf{0}
\end{aligned}
$$

Three types of Dantzig-Wolfe reformulation:

1. for Linear Programs (LP)
2. for (Mixed) Integer Programms (IP) by Convexification
3. for (Mixed) Integer Programms (IP) by Discretization

"Original model/formulation":

$$\begin{aligned} z_{IP} &= \min c^\top x \\ Ax &= b \\ Dx &= d \\ x &\geq \mathbf{0}, \quad \text{integer} \end{aligned}$$

$X = \{x : Dx = d, x \geq \mathbf{0} \ \text{ integer}\}$

$\{x_p, p \in P\}$ extreme points of $conv(X)$

$\{x_r, r \in R\}$ extreme rays of $conv(X)$

Substitute

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r$$

Equivalent IP (integer master program): *"extensive formulation"*

$$z_{IMP} = \min \sum_{p \in P} (c^\top x_p) \lambda_p + \sum_{r \in R} (c^\top x_r) \lambda_r$$

$$\begin{aligned} \sum_{p \in P} (Ax_p) \lambda_p + \sum_{r \in R} (Ax_r) \lambda_r &= b \qquad [\pi] \\ \sum_{p \in P} \lambda_p &= 1 \qquad [\mu] \end{aligned}$$

$$\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R$$

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r \quad \text{integer} \leftarrow \boxed{\text{Important!}}$$

D-W Reformulation for IP by Convexification

- Original and extensive **integer** formulations are equivalent!

$$z_{IP} = z_{IMP}$$

- Original and extensive **LP relaxations** are generally not equivalent!

$$z_{LP} \leq z_{MP}$$

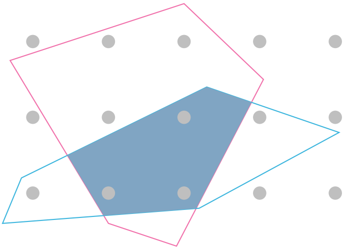
- Solution of MP by **column generation:**
  - $\rightarrow$ RMP solved with LP-solver (primal, dual, barrier)
    - $>$ dual solution π, μ
  - $\rightarrow$ pricing (=sub)problem (PP):

$$\begin{aligned} \tilde{c}^\star(\pi, \mu) \quad &= \quad -\mu + \min(c^\top - \pi^\top A)x \\ \text{s.t.} \quad & \quad Dx = d \\ & \quad x \geq \mathbf{0} \ \ \text{integer} \end{aligned}$$

    - $>$ (use algorithm that ensures) extreme point(s) x_p or extreme ray(s) x_r with negative rdc
    - $>$ partial pricing! ($\rightarrow$ heuristics, heuristics, heuristics)
- Solution of IMP by **branch-and-price:**
  - $\rightarrow$ Each node of the branch-and-bound tree is a (modified) MP solved by column generation
- Block-diagonal structure ...

D-W Reformulation – Better Bounds

Original Formulation: LP and IP

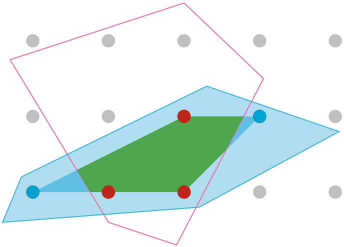


$\{x : Ax = b\}$

$\{Dx = d, x \geq \mathbf{0}\}$

$$P_{LP} = \{x : Ax = b\} \cap \{x : Dx = d, x \geq \mathbf{0}\}$$

Extensive Formulation: MP and IMP



$\{x : Ax = b\}$

$\{Dx = d, x \geq \mathbf{0}\}$

$conv(\{Dx = d, x \geq \mathbf{0} \text{ integer}\})$

$$P_{MP} = \{x : Ax = b\} \cap conv(\{Dx = d, x \geq \mathbf{0} \text{ integer}\})$$

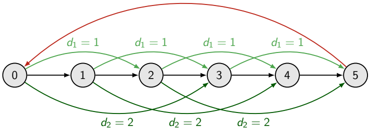
Better bound possible if $P_{LP} \supsetneq P_{MP}$!

Equality if subproblem formulation is integral.

Example: Cutting Stock Problem

Arc-flow formulation: Network with nodes $V = \{0, 1, 2, \ldots, L\}$ and

- arcs $(i, i + \ell_d) \in E$ representing piece d
- arcs $(i, i + 1) \in E$ representing slack
- one arc $(L, 0)$



Similar to [Valério de Carvalho, 1999]:

$$
\begin{aligned}
z_{IP} \quad = \quad & \min \sum_{(0,i)\in\delta^+(0)} x_{0i} \\
\text{s.t.} \quad & \sum_{(i,i+\ell_d)\in E} x_{i,i+\ell_d} \geq b_d \qquad d = 1, \ldots, m \\
& \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(h,i)\in\delta^-(i)} x_{hi} = 0 \qquad i \in V \\
& x_{ij} \geq 0, \quad \text{integer} \qquad (i,j) \in E
\end{aligned}
$$

D-W Reformulation – Cutting Stock

Example (cont'd): Cutting Stock Problem

$$
\begin{aligned}
z_{IP} \quad = \quad & \min \sum_{(0,i)\in\delta^+(0)} x_{0i} \\
\text{s.t.} \quad & \sum_{(i,i+\ell_d)\in E} x_{i,i+\ell_d} \geq b_d \qquad d = 1,\ldots,m \\
& \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(h,i)\in\delta^-(i)} x_{hi} = 0 \qquad i \in V \\
& x_{ij} \geq 0, \quad \text{integer} \qquad (i,j) \in E
\end{aligned}
$$

- **Pricing problem** constraints:
  - $\rightarrow$ $X = \{x \geq \mathbf{0} : \text{circulation}\}$
  - $\rightarrow$ homogeneous system, unbounded
- One extreme point $\mathbf{0}$ $\leftarrow$ $\boxed{\text{Not needed, cost 0, flow 0}}$
- All cycles are extreme rays

D-W Reformulation – Cutting Stock

Example (cont'd): Cutting Stock Problem

D-W reformulation of arc-flow formulation:

$$\begin{aligned} z_{IMP} = \quad \min \quad & \sum_{r \in R} \lambda_r \\ \text{s.t.} \quad & \sum_{r \in R} \left(\sum_{(i,i+\ell_d) \in E} (x_r)_{i,i+\ell_d} \right) \lambda_r \geq b_d \qquad d = 1, 2, \ldots, m \\ & \lambda_r \geq 0 \qquad r \in R \\ & x_{ij} = \sum_{r \in R} (x_r)_{ij} \lambda_r \quad \text{integer} \qquad (i,j) \in E \end{aligned}$$

Compare with **Modell, Gilmore & Gomory (1961, 1963)**

- Coefficient of λ_r in dth constraint:
  $\rightarrow$ number of times cycle/ray/pattern r uses piece d/arcs $(i, i + \ell_d)$
- Integer constraints on x_{ij}, not on λ_r!
- Pricing problem with integral formulation
  $\rightarrow$ equivalent LP relaxations, $z_{LP} = z_{MP}$

Example: Vehicle Routing Problem with Time Windows

Three-Index Formulation for the VRPTW:

$$
\begin{aligned}
z_{IP} \quad = \quad & \min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \\
\text{s.t.} \quad & \sum_{k \in K} \sum_{(i,j) \in \delta^+(i)} x_{ij}^k = 1 \quad \forall\, i \in N \\
& \sum_{(o,j) \in \delta^+(o)} x_{oj}^k = \sum_{(i,d) \in \delta^-(d)} x_{id}^k \leq 1 \quad \forall\, k \in K \\
& \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(j,i) \in \delta^-(i)} x_{ji}^k = 0 \quad \forall\, i \in N, k \in K \\
& u_i^k - u_j^k + Q x_{ij}^k \leq Q - q_j \quad \forall\, (i,j) \in A, k \in K \\
& q_i \leq u_i^k \leq Q \quad \forall\, i \in N, k \in K \\
& T_i^k - T_j^k + M_{ij} x_{ij}^k \leq M_{ij} - t_{ij} \quad \forall\, (i,j) \in A, k \in K \\
& a_i \leq T_i^k \leq b_i \quad \forall\, i \in N, k \in K \\
& x_{ij}^k \in \{0,1\} \quad \forall\, (i,j) \in A, k \in K
\end{aligned}
$$

- **Pricing problem** constraints:
  - $\rightarrow$ decompose into $|K|$ blocks (the *vehicles*)
  - $\rightarrow$ describe feasible vehicle routes ($\rightarrow$ extreme points: λ_p)
  - $\rightarrow$ no extreme rays

Example: (cont'd) Vehicle Routing Problem with Time Windows

D-W reformulation with blocks $k \in K$:

$$
\begin{aligned}
\min \quad & \sum_{k\in K} \sum_{(i,j)\in A} \sum_{p\in P} c_{ij} (x_p^k)_{ij} \lambda_p^k \\
\text{s.t.} \quad & \sum_{k\in K} \sum_{(i,j)\in \delta^+(i)} \sum_{p\in P} (x_p^k)_{ij} \lambda_p^k = 1 \qquad i \in N \qquad [\pi_i] \\
& \sum_{p\in P} \lambda_p^k = 1 \qquad k \in K \qquad [\mu_k] \\
& \lambda_p^k \geq 0 \qquad k \in K, p \in P \\
& x_{ij}^k = \sum_{p\in P} (x_p^k)_{ij} \lambda_p^k \in \{0,1\} \qquad k \in K, (i,j) \in A
\end{aligned}
$$

Let

c_p cost of route p: $\quad c_p = \sum_{(i,j)\in A} c_{ij} (x_p^k)_{ij}$

a_{ip} indicator if route p visits customer i: $\quad a_{ip} = \sum_{(i,j)\in \delta^+(i)} (x_p^k)_{ij}$

$a_{ij,p}$ indicator if route p uses arc (i,j): $\quad a_{ij,p} = (x_p^k)_{ij}$

D-W Reformulation – VRPTW

Example: (cont'd) Vehicle Routing Problem with Time Windows

(Nicer looking) D-W reformulation with blocks $k \in K$:

$$
\begin{aligned}
\min \quad & \sum_{k \in K} \sum_{p \in P} c_p \lambda_p^k \\
\text{s.t.} \quad & \sum_{k \in K} \sum_{p \in P} a_{ip} \lambda_p^k = 1 \qquad i \in N & [\pi_i] \\
& \sum_{p \in P} \lambda_p^k = 1 \qquad k \in K & [\mu_k] \\
& \lambda_p^k \geq 0 \qquad k \in K, p \in P \\
& x_{ij}^k = \sum_{p \in P} a_{ij,p} \lambda_p^k \in \{0, 1\} \qquad k \in K, (i,j) \in A
\end{aligned}
$$

- All blocks $k \in K$ are identical!

  $\rightarrow$ Aggregation! (eliminates symmetry w.r.t. vehicles)
  $\rightarrow$ $\lambda_p = \sum_{k \in K} \lambda_p^k \quad \geq 0$
  $\rightarrow$ $x_{ij} = \sum_{k \in K} x_{ij}^k \quad \in \mathbb{Z}_+$

Example: (cont'd) Vehicle Routing Problem with Time Windows

(Aggregated) D-W reformulation:

$$
\begin{aligned}
\min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} a_{ip} \lambda_p = 1 \qquad i \in N \qquad && [\pi_i] \\
& \sum_{p \in P} \lambda_p = |K| && [\mu] \\
& \lambda_p \geq 0 \qquad p \in P \\
& x_{ij} = \sum_{p \in P} a_{ij,p} \lambda_p \in \mathbb{Z}_+ \qquad (i,j) \in A
\end{aligned}
$$

- Here(!) we can replace integer by binary constraints
  - $\rightarrow$ $x_{ij} = \sum_{p \in P} a_{ij,p} \lambda_p \in \{0, 1\}$
  - $\rightarrow$ $x_{ij} \geq 2$ is always infeasible!
- Typically, the empty route is allowed:
  - $\rightarrow$ $\sum_{p \in P} \lambda_p \leq |K|$

Example: Vertex Coloring Problem

Given Graph $G = (V, E)$

Task Assign colors to vertices such that adjacent vertices receive different colors and the number of colors used is minimum

Sufficiently many colors $K = \{\textit{blue}, \textit{green}, \textit{orange}, \textit{magenta}, \dots\}$

$$
\begin{aligned}
z_{IP} = \min \quad & \sum_{k \in K} y^k \\
\text{s.t.} \quad & \sum_{k \in k} x_i^k = 1 && i \in V \\
& x_i^k + x_j^k \leq 1 && \{i, j\} \in E, k \in K \\
& x_i^k \leq y^k && i \in V, k \in K \\
& x_i^k, y^k \in \{0, 1\} && i \in V, k \in K
\end{aligned}
$$

D-W Reformulation – Vertex Coloring

Example (cont'd): Vertex Coloring Problem

$$
\begin{aligned}
z_{IP} = \min \quad & \sum_{k \in K} y^k \\
\text{s.t.} \quad & \sum_{k \in k} x_i^k = 1 && i \in V \\
& x_i^k + x_j^k \leq 1 && \{i, j\} \in E, k \in K \\
& x_i^k \leq y^k && i \in V, k \in K \\
& x_i^k, y^k \in \{0, 1\} && i \in V, k \in K
\end{aligned}
$$

- **Pricing problem** constraints:
  - → decompose into $|K|$ blocks (*the colors*)
  - → coupling of *"color and coloring"* and non-adjacency
- Extreme points:
  - → $(y^k, x_i^k) = \mathbf{0}$
  - → $(y^k, x_i^k) = (1, x_i) \in \{0, 1\}^{1+|V|}$, where $\{i \in V : x_i = 1\}$ are independent sets
    - > new variables λ_S^k for independent sets $S \subset V$
- No extreme rays

Example (cont'd): Vertex Coloring Problem

Let $\mathcal{S} = \{S \subseteq V : S$ independent set in $G\}$

D-W reformulation with blocks $k \in K$:

$$
\begin{aligned}
\min \quad & \sum_{k \in K} \sum_{S \in \mathcal{S}} \lambda_S^k \\
\text{s.t.} \quad & \sum_{k \in k} \sum_{S \in \mathcal{S} : i \in S} \lambda_S^k = 1 && i \in V \\
& \sum_{S \in \mathcal{S}} \lambda_S^k \leq 1 && k \in K && \text{(Why } \leq \text{?)} \\
& \lambda_S^k \geq 0 && k \in K, S \in \mathcal{S} \\
& x_i^k = \sum_{S \in \mathcal{S} : i \in S} \lambda_S^k \in \{0, 1\} && i \in V, k \in K
\end{aligned}
$$

- Integer requirement $x_i^k \in \{0, 1\}$ can be shifted on λ_S^k variables

Example (cont'd): Vertex Coloring Problem

Integer requirement shifted on λ_S^k

D-W reformulation with blocks $k \in K$:

$$
\begin{aligned}
\min \quad & \sum_{k\in K} \sum_{S\in\mathcal{S}} \lambda_S^k \\
\text{s.t.} \quad & \sum_{k\in k} \sum_{S\in\mathcal{S}:i\in S} \lambda_S^k = 1 && i \in V \\
& \sum_{S\in\mathcal{S}} \lambda_S^k \leq 1 && k \in K \\
& \lambda_S^k \in \{0,1\} && k \in K, S \in \mathcal{S}
\end{aligned}
$$

- All blocks $k \in K$ are identical!
  - $\rightarrow$ Aggregation! (eliminates symmetry w.r.t. colors)
  - $\rightarrow$ $\lambda_S = \sum_{k\in K} \lambda_S^k \in \mathbb{Z}_+$ for all $S \in \mathcal{S}$

D-W Reformulation – Vertex Coloring

Example (cont'd): Vertex Coloring Problem

Aggregated D-W reformulation:

$$
\begin{aligned}
z_{IMP}^{aggr} = \min \quad & \sum_{S \in \mathcal{S}} \lambda_S \\
\text{s.t.} \quad & \sum_{S \in \mathcal{S} : i \in S} \lambda_S = 1 && i \in V \\
& \sum_{S \in \mathcal{S}} \lambda_S \leq |K| && \text{(Redundant! Why?)} \\
& \lambda_S \geq 0 \quad \text{integer} && S \in \mathcal{S}
\end{aligned}
$$

- Here(!) we can replace integer by binary constraints $\lambda_S \in \{0, 1\}$

  $\rightarrow$ $\lambda_S \geq 2$ is always infeasible!

Three types of Dantzig-Wolfe reformulation:

1. for Linear Programs (LP)
2. for (Mixed) Integer Programms (IP) by Convexification
3. for (Mixed) Integer Programms (IP) by Discretization

D-W Reformulation for IP by Discretization

"Original model/formulation":

$$
\begin{aligned}
z_{IP} &= \min c^\top x \\
Ax &= b \\
Dx &= d \\
x &\geq \mathbf{0}, \quad \text{integer}
\end{aligned}
$$

$X = \{x : Dx = d, x \geq \mathbf{0} \ \text{ integer}\}$

$\{x_p, p \in P\}$ integer points of X

$\{x_r, r \in R\}$ integer rays of X

Substitute

$$x = \sum_{p \in P} \lambda_p x_p + \sum_{r \in R} \lambda_r x_r \quad \text{binary/non-neg. integer combinations}$$

Equivalent IP (integer master program): ***"extensive formulation"***

$$
\begin{aligned}
z_{IMP} = \min \sum_{p \in P} (c^\top x_p) \lambda_p + \sum_{r \in R} (c^\top x_r) \lambda_r \\
\sum_{p \in P} (A x_p) \lambda_p + \sum_{r \in R} (A x_r) \lambda_r &= b \qquad [\pi] \\
\sum_{p \in P} \lambda_p &= 1 \qquad [\mu] \\
\lambda_p \in \{0, 1\}, p \in P, \quad \lambda_r \in \mathbb{Z}_+, r \in R \\
x = \sum_{p \in P} \lambda_p x_p \quad + \sum_{r \in R} \lambda_r x_r \quad &\leftarrow \boxed{\text{redundant}}
\end{aligned}
$$

Kantorovich (1960) formulation for Cutting Stock:

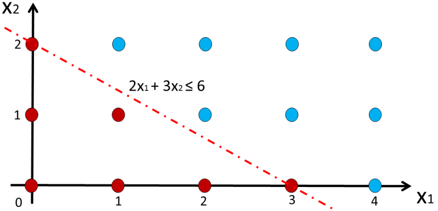
$$\begin{aligned}
z_{IP} \quad = \quad & \min \sum_{k \in K} y^k \\
\text{s.t.} \quad & \sum_{k \in K} x_i^k \geq b_i \qquad i = 1, 2, \ldots m \\
& \sum_{i=1}^{m} \ell_i x_i^k \leq L y^k \qquad k \in K \\
& y^k \in \{0, 1\} \qquad k \in K \\
& x_i^k \in \mathbb{Z}_+ \qquad k \in K, i = 1, 2, \ldots, m
\end{aligned}$$

Feasible integer points in the kth block:

$$S^k = \{(y^k, \mathbf{x}_i^k) \in \{0, 1\} \times \mathbb{Z}_+^m : \sum_{i=1}^{m} \ell_i x_i^k \leq L y^k\}$$

Example (taken from [Ben Amor and Valério de Carvalho, 2005]):

- Roll $L = 6$, $m = 2$ pieces of length $\ell_1 = 2$ and $\ell_2 = 3$
- Demand $b_1 = 4$ and $b_2 = 3$



Integer KP
cannot generate $(1, 1)$

$(1, 1)$ is convex combination of
$(3, 0)$, $(0, 2)$ and $(0, 0)$

Discretization uses

$$S^k = \{(0,(0,0)),(1,(0,0)),(1,(1,0)),(1,(0,1)),(1,(1,1)),(1,(2,0)),(1,(0,2)),(1,(3,0))\}$$

while Convexification uses

$$X^k = conv(S^k) = conv\left(\{(0,(0,0)),(1,(0,0)),(1,(0,2)),(1,(3,0))\}\right)$$

i.e., $P = \{(0,(0,0)),(1,(0,0)),(1,(3,0)),(1,(0,2))\}$ are the extreme points.

Convexification vs. Discretization

- **Integer formulations** are equivalent!

$$z_{IMP}^{conv} = z_{IMP}^{disc}$$

- **LP relaxations** are equivalent!

$$z_{MP}^{conv} = z_{MP}^{disc}$$

- Same pricing (=sub)problem (PP):

$$\begin{aligned} \tilde{c}^\star(\pi, \mu) \quad &= \quad -\mu + \min(c^\top - \pi^\top A)x \\ \text{s.t.} \quad & \quad Dx = d \\ & \quad x \geq \mathbf{0} \ \text{ integer} \end{aligned}$$

  $\rightarrow$ every interior point is dominated with respect to objective values by at least one extreme point of $conv(X)$

  $\rightarrow$ extreme point(s) x_p or extreme ray(s) x_r with negative rdc

- Interior points are not needed for LP relaxation
- Branching/cutting ensures that interior points are generated

  $\rightarrow$ structure of subproblem changes. . . !!

Some Takeaways

Why should we think about D-W Reformulation?

- For Linear Programs
  - $\rightarrow$ does (usually) not pay off: just solve original model
  - $\rightarrow$ not beneficial even for large LPs with block-diagonal structure
- For (Mixed) Integer Programms
  - $\rightarrow$ better dual bounds
  - $\rightarrow$ elimination of symmetry
  - $\rightarrow$ manage non-linearities in the subproblem
  - $\rightarrow$ for many applications: very powerful algorithms for subproblems available

PART III

Branching and Cutting

=

From integer formulations to integer solutions

Branch-Price-and-Cut

Solving integer formulations:

- Standard method for solving IP (*original formulation*): branch-and-cut
- Branch-price-and-cut is the equivalent for solving IMP (*extensive formulation*)

  $\rightarrow$ column generation is reapplied on modified D-W reformulation

  - at each node of the branch-and-bound search tree
  - after adding (violated) valid inequalities

We discuss:

1. Branching on the λ variables of the MP
2. Branching and cutting on the x variables of the original formulation
3. Ryan-Foster branching for set partitioning MPs
4. Valid inequalities on the λ variables of the MP

Branching on λ Variables

Do not branch (directly) on the λ variables of the master program!

1. Search tree tends to be very unbalanced
   - $\rightarrow$ branch $\lambda_p = 1$ is typically a very strong decision ($\rightarrow$ lower bound improves well)
   - $\rightarrow$ branch $\lambda_p = 0$ is typically a very weak decision ($\rightarrow$ MP almost unchanged, often lower bound don't improve)
2. Subproblem becomes much more involved!
   - $\rightarrow$ forbidden $p \in P$ must not be re-generated
   - $\rightarrow$ how to enforce this?

Branching and Cutting on x Variables

Original formulation with **additional constraints** in the x variables:

$$\begin{array}{rcll} z_{IP} & = & \min c^\top x & \\ Ax & = & b & \\ Ex & \leq & e & \leftarrow \quad \text{Additional constraints!} \\ Dx & = & d & \\ x & \geq & \mathbf{0}, & \text{integer} \end{array}$$

Two possibilities to integrate $Ex \leq e$ in D-W reformulation:

Master Constraints:

$$\begin{array}{rcll} Ax & = & b & \rightarrow \quad \text{new master} \\ Ex & \leq & e & \rightarrow \quad Ax = b \\ Dx & = & d & \end{array}$$

Subproblem Constraints:

$$\begin{array}{rcll} Ax & = & b & \\ Ex & \leq & e & \rightarrow \quad \text{new subproblem} \\ Dx & = & d & \rightarrow \quad Dx = d \end{array}$$

Integrating $Ex \leq e$ in Master Constraints:

Modified master program (additional constraints):

$$
\begin{array}{llll}
\min \sum_{p \in P} (c^\top x_p)\lambda_p + \sum_{r \in R} (c^\top x_r)\lambda_r & & & \\
\sum_{p \in P} (Ax_p)\lambda_p + \sum_{r \in R} (Ax_r)\lambda_r & = & b & [\pi] \\
\sum_{p \in P} (Ex_p)\lambda_p + \sum_{r \in R} (Ex_r)\lambda_r & \leq & e & [\alpha] \\
\sum_{p \in P} \lambda_p & = & 1 & [\mu] \\
\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R & & &
\end{array}
$$

Modified subproblem (modified reduced cost):

$$
\begin{array}{rcl}
\tilde{c}^\star(\pi, \alpha, \mu) & = & -\mu + \min(c^\top - \pi^\top A - \alpha^\top E)x \\
\text{s.t.} & & Dx = d \\
& & x \geq \mathbf{0} \quad \text{integer}
\end{array}
$$

Integrating $Ex \leq e$ in Subproblem Constraints:

Modified master program (modified columns):

$$
\begin{array}{lrcll}
\min & \sum_{p \in P_{(Ex \leq e)}} (c^\top x_p)\lambda_p + \sum_{r \in R_{(Ex \leq e)}} (c^\top x_r)\lambda_r & & & \\
 & \sum_{p \in P_{(Ex \leq e)}} (Ax_p)\lambda_p + \sum_{r \in R_{(Ex \leq e)}} (Ax_r)\lambda_r & = & b & [\pi] \\
 & \sum_{p \in P_{(Ex \leq e)}} \lambda_p & = & 1 & [\mu] \\
 & \lambda_p \geq 0, p \in P_{(Ex \leq e)}, \quad \lambda_r \geq 0, r \in R_{(Ex \leq e)} & & &
\end{array}
$$

Modified subproblem (additional constraints):

$$
\begin{array}{rcl}
\tilde{c}^\star(\pi, \mu) & = & -\mu + \min(c^\top - \pi^\top A)x \\
\text{s.t.} & & Ex \leq e \\
 & & Dx = d \\
 & & x \geq \mathbf{0} \quad \text{integer}
\end{array}
$$

Branching and Cutting on x Variables

Two possibilities. But which should we chose?

- Subproblem:
  - $\rightarrow$ convexifying the constraints is potentially stronger
  - $\rightarrow$ if they go well with (or even simplify) the subproblem algorithm
- Master:
  - $\rightarrow$ if the constraints link several variable types/subproblems
  - $\rightarrow$ if subproblem algorithm is not compatible with them

Examples: (for VRPs)

Additional Master constraints

- branching on # of vehicles
  $\sum_{(0,j)} x_{0j} \leq k$ or
  $\sum_{(0,j)} x_{0j} \geq k+1$
- cut on total cost
  $\sum_{(i,j)} c_{ij} x_{ij} \geq \lfloor LB \rfloor$
- 2-path cuts
  $\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq 2$
- branching on flow into subset
  $\sum_{(i,j)\in\delta^-(S)} x_{ij} \leq k$ or
  $\sum_{(i,j)\in\delta^-(S)} x_{ij} \geq k+1$

Additional Subproblem constraints

- branching on arcs
  $x_{ij} = 0$ or $x_{ij} = 1$
- branching on time vars
  TW $[a_i, m]$ or TW $[m+1, b_i]$
- branching on resource vars
  res in $[a_i, m]$ or res in $[m+1, b_i]$

Ryan-Foster Branching for Set Partitioning MPs

Assumptions:

- MP has set partitioning (sub)structure with rows $i \in I$
  - $\rightarrow$ Tasks $i \in I$ that have to be fulfilled (color vertices, pack items, visit customers,…)
  - $\rightarrow$ Aggregation has been performed on MP
- $x_i \in \{0,1\}$ indicator variable for task $i \in I$ in original model
- $x_i(p) \in \{0,1\}$ indicator if task i is covered by column $p \in P$

Property [Ryan and Foster, 1981]:

Let $\lambda = (\lambda_p)_{p \in P}$ be a fractional solution of a set partitioning model.
There exist two rows i and j with

$$w_{ij}(\lambda) := \sum_{p \in P:\ x_i(p) = x_j(p) = 1} \lambda_p \in (0,1) \qquad \leftarrow \qquad \textbf{fractional}$$

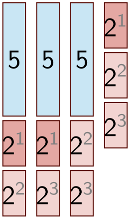
Branch on $w_{ij} = 0$ (=*separate*-branch) and $w_{ij} = 1$ (=*together*-branch)!

Ryan-Foster Branching for Set Partitioning MPs

Ryan and Foster branching for Bin Packing: Decide whether two items are packed together into one bin or packed separately into two bins.

Example: Length $L = 10$, $m = 4$ with $\ell_1 = 5, \ell_2 = \ell_3 = \ell_4 = 2$.

$$
\begin{aligned}
z(P_{BP}) \quad &= \quad \min \mathbf{1}^\top x \\
\text{s.t.} \quad & \quad \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} x \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& \quad \lambda \geq 0
\end{aligned}
$$



- P_{BP} has solution $\bar{\lambda} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})^\top$ with $z(P_{BP}) = \frac{4}{3} = 1.\overline{3}$
- For $i = 1$ (weight 5) and $j = 2$ (weight 2), the solution $\bar{\lambda}$ implies that that "in $w_{ij} = 2/3$ of the cases", they are packed together
- Two branches:

  $\rightarrow$ together: items $i = 1$ and $j = 2$ have to be in the same bin
  $\rightarrow$ separate: items $i = 1$ and $j = 2$ have to be in different bins

Ryan-Foster Branching for Set Partitioning MPs

Examples:

Bin Packing	items i and j are packed in different bins/the same bin
Vertex Coloring	vertices i and j receive different colors/the same color
VRPs	customers i and j are visited by different vehicles/the same vehicle

How to handle R-F branching decisions?

- Master problem:

  $\rightarrow$ remove incompatible columns

- Subproblem: (formally)

  $\rightarrow$ integrate additional $w_{ij} \in \{0,1\}$ variables
  $\rightarrow$ ensure $w_{ij} = 1$ if and only if $x_i = x_j$

Ryan-Foster Branching for Set Partitioning MPs

Depending on the type of subproblem (or the solution algorithm), the branches $w_{ij} = 0$ and $w_{ij} = 1$ may be **simple or hard** to implement:

- Binary knapsack problems ($\rightarrow$ bin packing): **hard**
  - $\rightarrow$ merge items ($w_{ij} = 1$) or add conflict ($w_{ij} = 0$)
  - $\rightarrow$ knapsack problem with (general) conflict constraints is hard
- Independent set problems ($\rightarrow$ vertex coloring): **simple**
  - $\rightarrow$ merge vertices ($w_{ij} = 1$) or add edge ($w_{ij} = 0$)
- ESPPRCs solved by labeling ($\rightarrow$ VRPs): **hard**
  - $\rightarrow$ pairing ($w_{ij} = 1$) or anti-pairing ($w_{ij} = 0$) require additional resource
- Formulation directly solved with MIP solver: **simple**
  - $\rightarrow$ set $w_{ij} = 1$ or $w_{ij} = 0$ if w variables are present
  - $\rightarrow$ otherwise: add constraints $x_i = x_j$ or $x_i + x_j \leq 1$

Valid inequalities on λ Variables

Master program with additional constraints in the λ variables:

$$
\begin{aligned}
\min \sum_{p \in P} (c^\top x_p)\lambda_p + \sum_{r \in R} (c^\top x_r)\lambda_r & & & \\
\sum_{p \in P} (A x_p)\lambda_p + \sum_{r \in R} (A x_r)\lambda_r &= b & & [\pi] \\
\sum_{p \in P} g(x_p)\lambda_p + \sum_{r \in R} g(x_r)\lambda_r &\leq g & & [\gamma] \\
\sum_{p \in P} \lambda_p &= 1 & & [\mu] \\
\lambda_p \geq 0, p \in P, \quad \lambda_r \geq 0, r \in R & & &
\end{aligned}
$$

- $g(x_p)/g(x_r)$: are the cut coefficients of λ_p/λ_r in the master

Valid inequalities on λ Variables

Corresponding subproblem: (master with add. constraints in λ)

$$\begin{aligned} \tilde{c}^\star(\pi,\alpha,\mu) \quad &= \quad -\mu + \min(c^\top - \pi^\top A)x - \gamma^\top g(x) \\ \text{s.t.} \quad & \quad Dx = d \\ & \quad x \geq \mathbf{0} \quad \text{integer} \end{aligned}$$

- $g(x_p)/g(x_r)$ are the cut coefficients of λ_p/λ_r in the master
- Subproblem needs to compute these coefficients
- $g(x)$ is a *function* in x
  - $\rightarrow$ does generally not translate back to (individual) original variables x
    - $>$ changes structure of subproblem
  - $\rightarrow$ compare with coefficient of constraints in x variables: Ex_p/Ex_r

Valid inequalities on λ Variables

Example: Subset Row Inequalities (w. subsets of size 3) [Jepsen et al., 2008]

Typical situation for set partitioning:

$$\begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \iddots \\ \cdots & 1 & 1 & 0 & \cdots \\ \cdots & 1 & 0 & 1 & \cdots \\ \cdots & 0 & 1 & 1 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix} \quad \begin{array}{l} \\ \leftarrow \text{task/row } i_1 \\ \leftarrow \text{task/row } i_2 \\ \leftarrow \text{task/row } i_3 \\ \\ \end{array}$$

with fractional solution $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$!

This solution can be cut by inequality

$$\lambda_1 + \lambda_2 + \lambda_3 \leq 1$$

General form:

$$\sum_{p \in P_S} \lambda_p \leq 1$$

- S subset of tasks with $|S| = 3$
- P_S subset of columns that cover at least two tasks from S

  $\rightarrow$ i.e., $g(x_p) = \begin{cases} 1 & \text{if column covers two or more tasks of } S \\ 0 & \text{otherwise} \end{cases}$

Valid inequalities on λ Variables

Example: Chvátal-Gomory Rank-1 Cuts

Assumptions:

- Master program is (extended) set partitioning/packing formulation
- Tasks/rows $i \in I$
- Coefficient a_{ip} of column p in row i
- Weights $u_i \in [0, 1)$

Chvátal-Gomory Rank-1 Cut:

$$\sum_{p \in P} \left\lfloor \sum_{i \in I} u_i a_{ip} \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in I} u_i \right\rfloor$$

Remarks:

- Coefficient $g(x_p) = \lfloor \sum_{i \in I} u_i a_{ip} \rfloor$ must be computed when solving the subproblem
- Subset row inequalities are Chvátal-Gomory rank-1 cuts

Some Takeaways

- Never ever ever ever branch directly on master variables!
  - $\rightarrow$ there might be exceptions. . . ?!
- Put valid inequalities in the subproblem, if they go well with it
- Valid inequalities in the master:
  - $\rightarrow$ inequalities in the original x variables (= *robust cuts*)
    - $>$ many families of inequalities known in the literature
    - $>$ often already implied by the reformulation
    - $>$ impact on subproblem: only reduced cost change, structure does not change
      ($\rightarrow$ there might be exceptions: linear node costs)
  - $\rightarrow$ inequalities in the λ variables (= *non-robust cuts*)
    - $>$ might be stronger
    - $>$ impact on subproblem: structural change
      - · harder to solve/adaptation of algorithm necessary
      - · add cautiously
- Number of branch-and-bound nodes that can be explored is typically limited (compared to branch-and-cut)

Hatem Ben Amor and José Valério de Carvalho. Cutting stock problems. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 131–161. Springer US, 2005.

Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497—511, 2008.

D. M. Ryan and B. A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer scheduling of public transport: Urban passenger vehicle and crew scheduling*, pages 269–280. Amsterdam, North-Holland, 1981.

J. M. Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, 1999.