Universidade do Minho

Decomposition Methods for Integer Programming

J.M. Valério de Carvalho
vc@dps.uminho.pt

Departamento de Produção e Sistemas
Escola de Engenharia, Universidade do Minho
Portugal

PhD Course
Programa Doutoral em Engenharia Industrial e de Sistemas (PDEIS)
Universidade do Minho
2010

Global outline

- Part I - Decomposition methods
- Part II - Applications
- Part III - Branch-and-price algorithms
- Part IV - Branch-and-price algorithms (cont.)
- Part V - Stabilization
- Part VI - Practical issues, accelerating strategies and heuristics
- Part VII - Primal cutting planes
- Part VIII - Heuristics
- Bibliography

Decomposition Methods

Outline

- Strength of models in integer programming (IP)
- Dantzig-Wolfe decomposition (DW)
- Comparative strength of IP, DW and LP
- Application Example
- Lagrangean relaxation vs. DW decomposition

Integer programming: strength of models

Integer Programming Problems (IP):

$$\begin{aligned} z_{IP} = \min \quad & cx \\ subj.\ to \quad & Ax = b \\ & x \geq 0 \text{ and integer} \end{aligned}$$

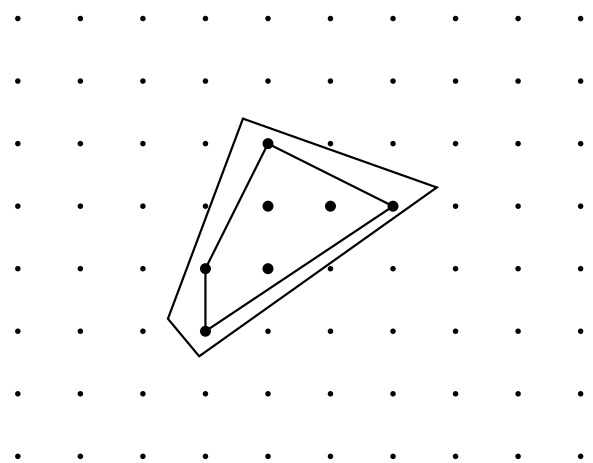
can be solved by branch-and-bound using the Linear Programming (LP) relaxation that results from relaxing the integrality conditions:

$$\begin{aligned} z_{LP} = \min \quad & cx \\ subj.\ to \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Crucial issue: some IP models are stronger, because their LP relaxations:

- provide closer description of convex hull of valid integer solutions.
- have LP optimal solution values closer to IP optimal solution values (smaller gap).

Motivation for branch-and-price



Some strong IP models have an exponential number of variables.

Solve them combining **column generation** and **branch-and-bound**.

Dantzig-Wolfe decomposition

May provide strong models (stronger than plain LP relaxation)...
... with an exponential number of variables.

$$\begin{array}{ll} \min & cx \\ suj. & Ax = b \\ & x \in X \\ & x \geq 0 \text{ and integer} \end{array}$$

Constraints decomposed in two sets:

- first set: general constraints → **Master Problem**.
- second set: constraints with special structure → **Subproblem**

Subproblem must be amenable for separate solution.

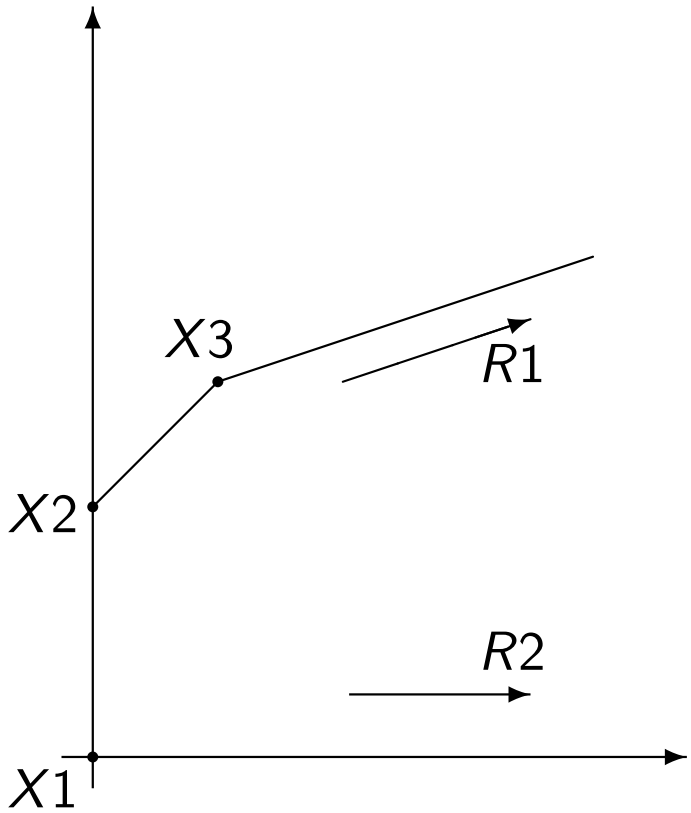
Dantzig-Wolfe decomposition: representation of a point

$$\begin{array}{ll} \min & cx \\ suj. & Ax = b \\ & \boxed{x \in X} \\ & x \geq 0 \end{array}$$

- Polyhedron X has I extreme points, denoted as $X_1, X_2, \ldots, X_I$, and K extreme rays, denoted as $R_1, R_2, \ldots, R_K$.
- Any point $x \in X$ is expressed as a convex combination of the extreme points of X plus a non-negative combination of the extreme rays of X :

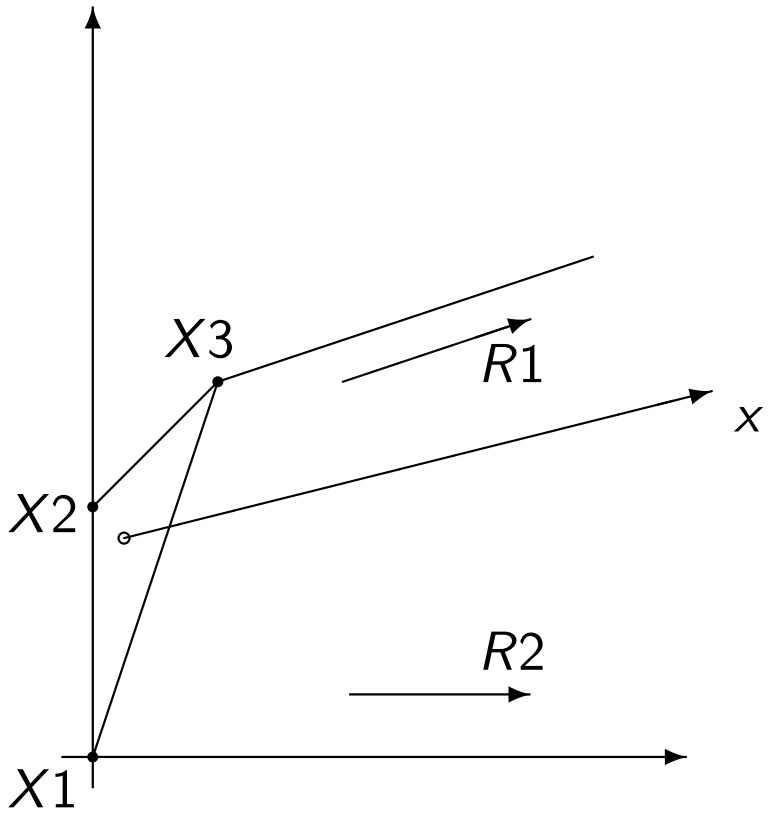
$$X = \left\{ x = \sum_{i=1}^{I} \lambda_i X_i + \sum_{k=1}^{K} \mu_k R_k, \sum_{i=1}^{I} \lambda_i = 1, \lambda_i \geq 0, \forall i, \mu_k \geq 0, \forall k \right\}$$

Dantzig-Wolfe decomposition: graphical representation



X_1, X_2 and X_3 are extreme points, and R_1 and R_2 are extreme rays. Valid space is unbounded.

Dantzig-Wolfe decomposition: graphical representation



x is expressed as a convex combination of X_1, X_2 and X_3 plus a non-negative combination of R_1 and R_2.

Some rewriting work...

replacing x in $\min\{cx : Ax = b, x \in X, x \geq 0\}$, we obtain

$$
\begin{aligned}
\min \quad & c\Big(\sum_{i=1}^{I} \lambda_i X_i + \sum_{k=1}^{K} \mu_k R_k\Big) \\
\textit{suj.} \quad & A\Big(\sum_{i=1}^{I} \lambda_i X_i + \sum_{k=1}^{K} \mu_k R_k\Big) = b \\
& \sum_{i=1}^{I} \lambda_i = 1 \\
& \lambda_i \geq 0, \forall i \\
& \mu_k \geq 0, \forall k
\end{aligned}
$$

Reformulation of the problem: master problem

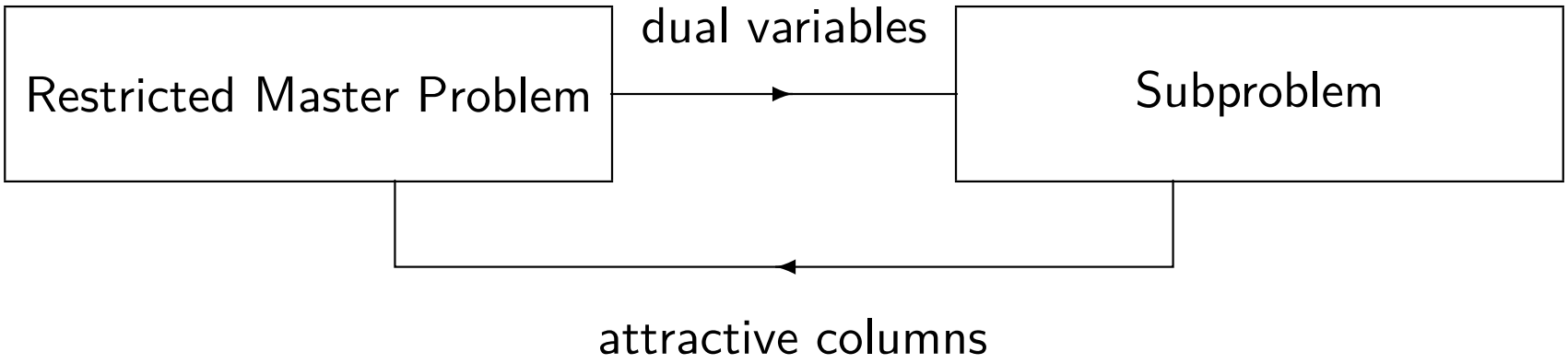
$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} (cX_i)\lambda_i + \sum_{k=1}^{K} (cR_k)\mu_k \\
\textit{subj. to} \quad & \sum_{i=1}^{I} (AX_i)\lambda_i + \sum_{k=1}^{K} (AR_k)\mu_k = b \\
& \sum_{i=1}^{I} \lambda_i = 1 \\
& \lambda_i \geq 0, \forall i \\
& \mu_k \geq 0, \forall k
\end{aligned}
$$

Decision variables: λ_i and μ_k.
Reformulated model is equivalent to original model.
Number of extreme points and extreme rays can be exponentially large.
Use column generation!

Column generation

Solve linear programming relaxation using column generation:

Choose an initial restricted set of columns
While (there is a column with negative reduced cost) do
    add column to restricted problem
    reoptimize
End While



[Dantzig, Wolfe, 1960; Ford, Fulkerson, 1958]

Integrality property

If X does not have the integrality property, the reformulated model is stronger than the linear programming relaxation.

Instead of searching extreme points and extreme rays in:

$$x \in Conv\{x \in X\},$$

search in:

$$x \in Conv\{x \in X \text{ and integer}\}.$$

That may not be too hard: in the Cutting Stock Problem, we have to find an integer solution of the subproblem (knapsack problem).

Comparative strengths of 3 different models:

- Integer programming model (IP)
- Linear programming relaxation model (LP)
- Dantzig-Wolfe decomposition model (DW)

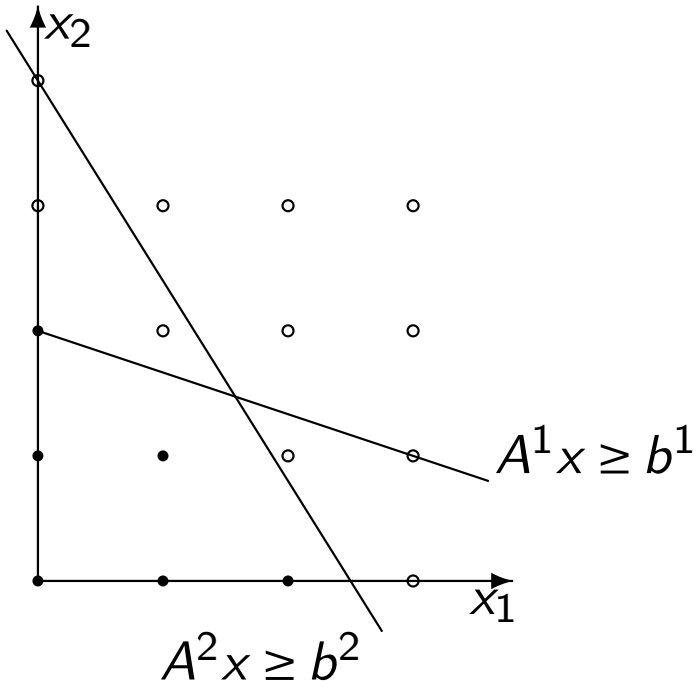
Three different models: IP, LP, DW

$$
\begin{aligned}
z_{IP} = \min \quad & cx \\
subj.\ to \quad & Ax = b \\
& x \in X \\
& x \geq 0 \text{ and integer}
\end{aligned}
$$

$$
\begin{aligned}
z_{LP} = \min \quad & cx \\
subj.\ to \quad & Ax = b \\
& x \in X \\
& x \geq 0
\end{aligned}
$$

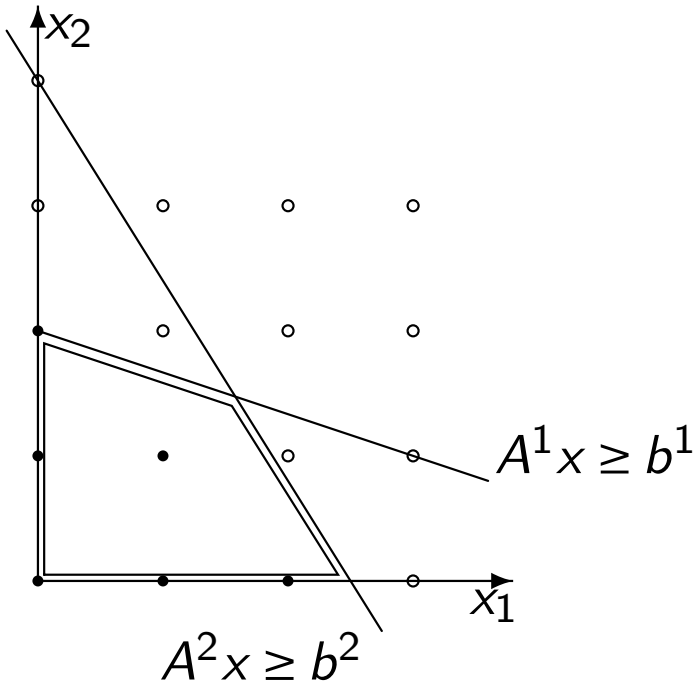
$$
\begin{aligned}
z_{DW} = \min \quad & cx \\
subj.\ to \quad & Ax = b \\
& x \in Conv\{x \in X \text{ and integer}\} \\
& x \geq 0
\end{aligned}
$$

Integer Problem: domain is a finite set of points



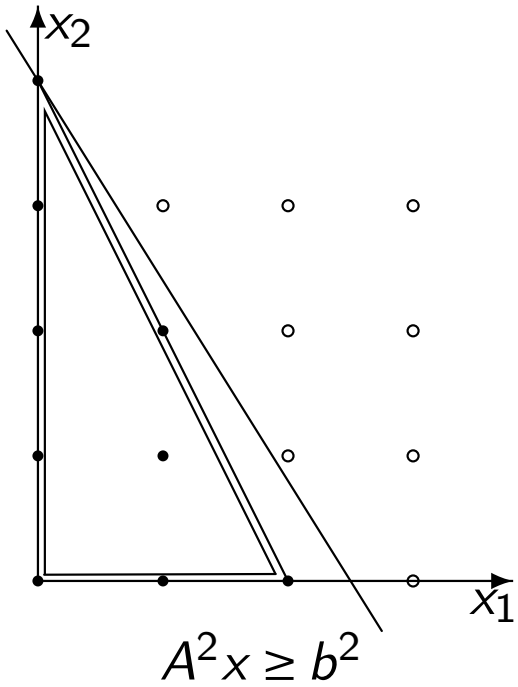
$$
\begin{aligned}
z_{IP} \quad &= \quad \min cx \\
subj.\ to \quad & \quad A^1x \geq b^1 \\
& \quad A^2x \geq b^2 \\
& \quad x \geq 0 \text{ and integer}
\end{aligned}
$$

Linear programming relaxation



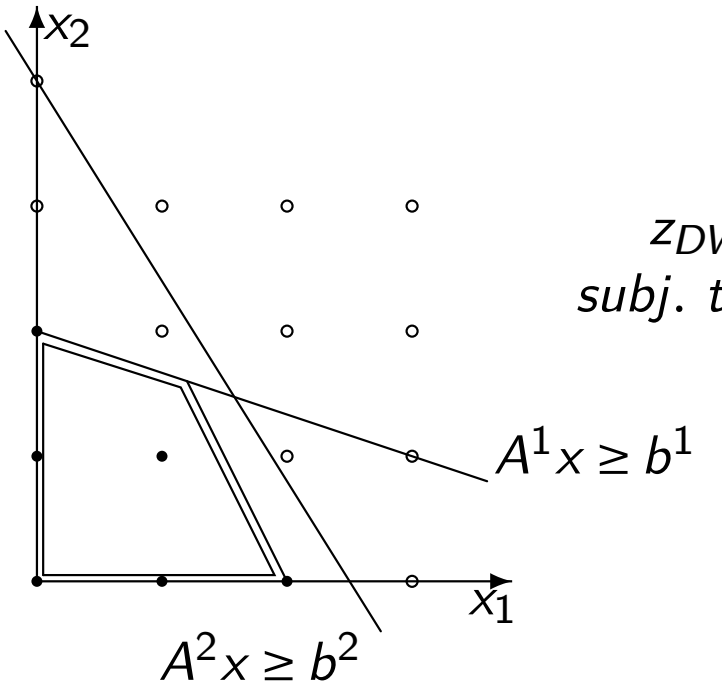
$$
\begin{array}{rcl}
z_{LP} & = & \min cx \\
subj.\ to & & A^1x \geq b^1 \\
 & & A^2x \geq b^2 \\
 & & x \geq 0
\end{array}
$$

$A^2x \geq b^2$ does not have the integrality property



$x \in Conv\{A^2x \geq b^2 \text{ and integer}\}$

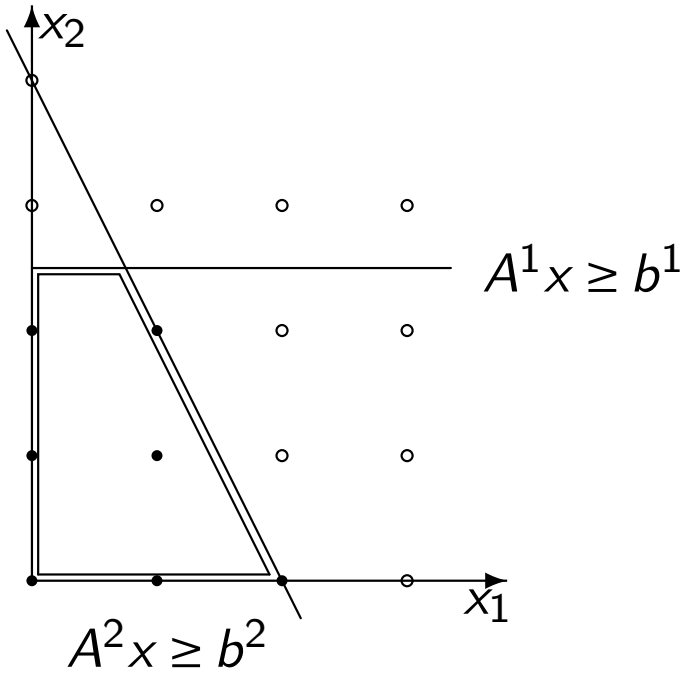
Reformulated model



$$
\begin{array}{rcl}
z_{DW} & = & \min cx \\
subj.\ to & & A^1x \geq b^1 \\
 & & x \in Conv\{A^2x \geq b^2 \text{ and integer}\} \\
 & & x \geq 0
\end{array}
$$

Reformulated model is stronger than LP relaxation.

If X is an integer polytope $\Rightarrow$ same bound as LP



$z_{LP} = z_{DW}$, because $X = Conv\{x \in X \text{ and integer}\}$ (compare models presented before).

Example

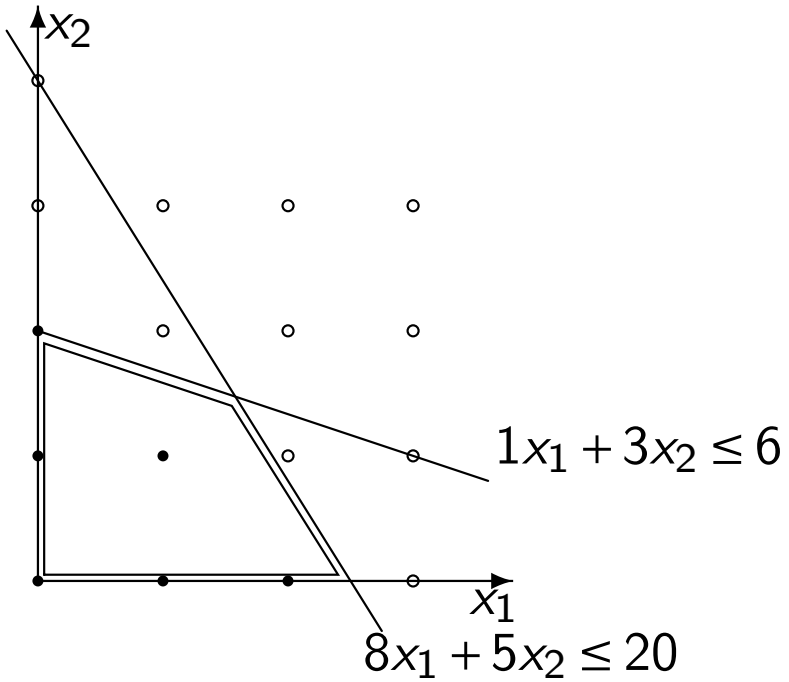
Integer programming:

- $\max\{x_1 + x_2 : 8x_1 + 5x_2 \leq 20, x_1 + 3x_2 \leq 6, x1, x2 \geq 0$ and integer$\}$.
- Several alternative integer optimal solutions: $(x_1, x_2) = (2,0), (0,2)$ and $(1,1)$ have objective function value $z_{IP} = 2$.

LP relaxation:

- LP relaxation optimal solution is $(x_1, x_2) = (30/19, 28/19)$, with objective function value $z_{LP} = 58/19 = 3.053$.
- Integrality gap $z_{LP} - z_{IP}$ equal to 1.053.

Integer and linear programming relaxation domains



$$
\begin{aligned}
z_{IP} &= \max 1x_1 + 1x_2 \\
\textit{subj. to} \quad & 1x_1 + 3x_2 \leq 6 \\
& 8x_1 + 5x_2 \leq 20 \\
& x_1, x_2 \geq 0, \text{ integer}
\end{aligned}
$$

Dantzig-Wolfe decomposition: $Conv\{x \in X \text{ and integer}\}$

$$Conv\{x \in X \text{ and integer}\} = Conv\{(x_1, x_2) : 8x_1 + 5x_2 \leq 20, x1, x2 \geq 0 \text{ and integer}\}$$

The integer extreme points of set X are:

$$X_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, X_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, X_3 = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$$

and the corresponding polytope is:

$$Conv\{x \in X \text{ and integer}\} = \{x \in \mathbb{R}^2 : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda_1 \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \lambda_3 \begin{pmatrix} 0 \\ 4 \end{pmatrix},$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \lambda_1, \lambda_2, \lambda_3 \geq 0\}.$$

Dantzig-Wolfe decomposition: reformulation

$$
\begin{array}{lrrcll}
\max & 1x_1 & +1x_2 & & & \\
subj.to & 1x_1 & +3x_2 & \leq & 6 & \text{(master problem)} \\
 & 8x_1 & +5x_2 & \leq & 20 & \text{(subproblem)} \\
 & x_1, & x_2 \geq 0 \text{ and integer} & & &
\end{array}
$$

$c = (1,1)$, $A^1 = (1,3)$, $A^2 = (8,5)$, $b^1 = 6$ and $b^2 = 20$, for the extreme points, we get:

$$cX_1 = \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0 \qquad A^1X_1 = \begin{bmatrix} 1 & 3 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$$cX_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 2 \\ 0 \end{bmatrix} = 2 \qquad A^1X_2 = \begin{bmatrix} 1 & 3 \end{bmatrix} * \begin{bmatrix} 2 \\ 0 \end{bmatrix} = 2$$

$$cX_3 = \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 4 \end{bmatrix} = 4 \qquad A^1X_3 = \begin{bmatrix} 1 & 3 \end{bmatrix} * \begin{bmatrix} 0 \\ 4 \end{bmatrix} = 12$$

Reformulated model (bounded case):

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{I} (cX_i)\lambda_i \\
\textit{subj. to} \quad & \sum_{i=1}^{I} (AX_i)\lambda_i \leq b \\
& \sum_{i=1}^{I} \lambda_i = 1 \\
& \lambda_i \geq 0, \forall i
\end{aligned}
$$

$$
\begin{array}{lrrrcl}
\max z_{DW} = & 0\lambda_1 & +2\lambda_2 & +4\lambda_3 & & \\
\textit{subj. to} & 0\lambda_1 & +2\lambda_2 & +12\lambda_3 & \leq & 6 \\
& \lambda_1 & +\lambda_2 & +\lambda_3 & = & 1 \\
& \lambda_1, & \lambda_2, & \lambda_3 \geq 0 & &
\end{array}
$$

Solution of complete model with simplex method

Notice that the order of the columns was changed, in order to have the identity matrix in the last two columns.

	z_{DW}	λ_2	λ_3	s_1	λ_1	
s_1	0	2	12	1	0	6
λ_1	0	1	1	0	1	1
	1	-2	-4	0	0	0

	z_{DW}	λ_2	λ_3	s_1	λ_1	
λ_3	0	$1/6$	1	$1/12$	0	$1/2$
λ_1	0	$5/6$	0	$-1/12$	1	$1/2$
	1	$-4/3$	0	$1/3$	0	2

	z_{DW}	λ_2	λ_3	s_1	λ_1	
λ_3	0	0	1	$1/10$	$-1/5$	$2/5$
λ_2	0	1	0	$-1/10$	$6/5$	$3/5$
	1	0	0	$1/5$	$8/5$	$14/5$

Optimal solution

- $\lambda_2 = 3/5$ and $\lambda_3 = 2/5$, with objective function value $14/5$.
- Optimal solution in the original space is a convex combination of the extreme points X_2 e X_3 with weights λ_2 e λ_3 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^* = 0\begin{pmatrix} 0 \\ 0 \end{pmatrix} + 3/5\begin{pmatrix} 2 \\ 0 \end{pmatrix} + 2/5\begin{pmatrix} 0 \\ 4 \end{pmatrix} = \begin{pmatrix} 6/5 \\ 8/5 \end{pmatrix}$$

- Integrality gap $z_{DW} - z_{IP}$ reduced to $14/5 - 2 = 0.8$ (stronger model).

A review of Linear Programming

Linear Programming (primal) maximization problem:

$$\begin{array}{ll} \max & cx \\ s.to & Ax \leq b \\ & x \geq 0 \end{array}$$

where $x \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$.
In matrix form:

A	I	b
$-c$	0	0

A basic solution (basis) is characterized by a set of m linearly independent columns, which form matrix $B \in \mathbb{R}^{m \times m}$.
$c_B \in \mathbb{R}^m$: vector of cost coefficients of the columns of B.

Exchange of Basis

Pre-multiplying B and c_B, the "matrix operator" performs an exchange of basis:

$$\begin{array}{|c|c|} \hline B^{-1} & 0 \\ \hline c_B B^{-1} & 1 \\ \hline \end{array} * \begin{array}{|c|} \hline B \\ \hline -c_B \\ \hline \end{array} = \begin{array}{|c|} \hline I \\ \hline 0 \\ \hline \end{array}$$

For the entire tableau:

$$\begin{array}{|c|c|} \hline B^{-1} & 0 \\ \hline c_B B^{-1} & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline A & I & b \\ \hline -c & 0 & 0 \\ \hline \end{array} =$$

$$= \begin{array}{|c|c|c|} \hline B^{-1}A & B^{-1} & B^{-1}b \\ \hline c_B B^{-1} A - c & c_B B^{-1} & c_B B^{-1} b \\ \hline \end{array}$$

Dual Problem and Optimality Conditions

$\min\{yb : yA \geq c, y \geq 0\}$ is the dual problem of $\max\{cx : Ax \leq b, x \geq 0\}$.

$y = c_B B^{-1}$ is a dual solution, which is valid when dual constraints are obeyed, *i.e.*:

- $yA \geq c \quad \equiv \quad c_B B^{-1} A - c \geq 0$
- $y \geq 0 \quad \equiv \quad c_B B^{-1} \geq 0$

Optimality conditions: if

- solution is dual valid (as shown above),
- solution is primal valid: $x = B^{-1} b \geq 0$, and
- complementary slackness conditions are obeyed, then
- finite optimal solution has value $c_B B^{-1} b$.

Example

	z_{DW}	λ_2	λ_3	s_1	λ_1	
s_1	0	2	12	1	0	6
λ_1	0	1	1	0	1	1
z_{DW}	1	-2	-4	0	0	0

Matrix B and corresponding cost vector c_B :

$$B = \begin{array}{cc} \lambda_3 & \lambda_1 \\ \hline 12 & 0 \\ 1 & 1 \end{array}$$

$$-c_B = \begin{array}{cc} & \\ \hline -4 & 0 \end{array}$$

Exchange of basis:

$$\begin{array}{|rrc|} \hline 1/12 & 0 & 0 \\ -1/12 & 1 & 0 \\ \hline 1/3 & 0 & 1 \\ \hline \end{array} * \begin{array}{|cc|} \hline 12 & 0 \\ 1 & 1 \\ \hline -4 & 0 \\ \hline \end{array} = \begin{array}{|cc|} \hline 1 & 0 \\ 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array}$$

Initial and final basis

	z_{DW}	λ_2	λ_3	s_1	λ_1	
s_1	0	2	12	1	0	6
λ_1	0	1	1	0	1	1
	1	-2	-4	0	0	0

	z_{DW}	λ_2	λ_3	s_1	λ_1	
λ_3	0	1/6	1	1/12	0	1/2
λ_1	0	5/6	0	-1/12	1	1/2
	1	-4/3	0	1/3	0	2

B^{-1}	0
$c_B B^{-1}$	1

$*$

A	I	b
$-c$	0	0

$=$

$=$

$B^{-1}A$	B^{-1}	$B^{-1}b$
$c_B B^{-1} A - c$	$c_B B^{-1}$	$c_B B^{-1} b$

Examples: $B^{-1}b$, $B^{-1}A_{X_2}$ and $c_B B^{-1}A_{X_2} - c_{X_2}$

$$B^{-1}b = \begin{bmatrix} 1/12 & 0 \\ -1/12 & 1 \end{bmatrix} * \begin{bmatrix} 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

$$B^{-1}A_{X_2} = \begin{bmatrix} 1/12 & 0 \\ -1/12 & 1 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/6 \\ 5/6 \end{bmatrix}$$

$$c_B B^{-1}A_{X_2} - c_{X_2} = \begin{bmatrix} 1/3 & 0 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \end{bmatrix} - 2 = -4/3$$

Solution of model with column generation

In this small example, it is easy to describe $Conv\{x \in X \text{ and integer}\}$ using only linear inequalities, and to obtain a polytope with integer extreme points:

$$\begin{aligned} Conv\{x \in X \text{ and integer}\} &= Conv\{(x_1, x_2) : 8x_1 + 5x_2 \leq 20, x1, x2 \geq 0 \text{ and intege} \\ &= \{(x_1, x_2) : 2x_1 + 1x_2 \leq 4, x1, x2 \geq 0\} \end{aligned}$$

Unfortunately, in general, for integer programming problems, that may not be easy (exponential number of constraints needed).

Subproblem: pricing extreme points of $Conv\{X\}$ out of the restricted master problem:

Reduced cost of $X_j : c_B B^{-1} A_j - c_j$ (maximization problem)
Column is attractive if its reduced cost < 0

Expression of reduced cost - I

$$
\begin{array}{lrrrrrl}
\max z_{DW} = & 0\lambda_1 & +2\lambda_2 & +4\lambda_3 & & & \text{(dual variable)} \\
\textit{subj. to} & 0\lambda_1 & +2\lambda_2 & +12\lambda_3 & \leq & 6 & (\pi_1) \\
 & \lambda_1 & +\lambda_2 & +\lambda_3 & = & 1 & (u) \\
 & \lambda_1, & \lambda_2, & \lambda_3 \geq 0 & & &
\end{array}
$$

Dual values of reformulated model: $c_B B^{-1} = \boxed{\pi_1 \quad u}$

Structure of columns in reformulated model (main body of tableau):

- upper part: results from changing variables in DW decomposition.
- lower part: convexity constraint ($\sum \lambda = 1$)

$$A_j = \boxed{\begin{matrix} A^1 X_j \\ 1 \end{matrix}}$$

Structure of columns in reformulated model (objective function):

$$c_j = cX_j$$

Expression of reduced cost - II

By substitution, we obtain:

Reduced cost of column of the reformulated model expressed in terms of original variables:

$$\begin{aligned} c_B B^{-1} A_j - c_j &= \pi_1 A^1 X_j + u * 1 - c X_j = \\ &= (\pi_1 A^1 - c) X_j + u \end{aligned}$$

Transferral of dual information to subproblem:

- we are now able to calculate the value of each point in $Conv\{X\}$
- look for the most attractive point (should be an extreme point)
- optimize the subproblem.

Insight: in the subproblem we have an objective function with:

- cost coefficients of the original variables x_1 and x_2 given by elements of the vector $(\pi_1 A^1 - c)$
- a constant term u

Solution 1 (starting solution)

	z_{DW}	s_1	λ_1	
s_1	0	1	0	6
λ_1	0	0	1	1
	1	0	0	0

Dual solution: $\pi_1 = 0, u = 0$

Reduced cost:

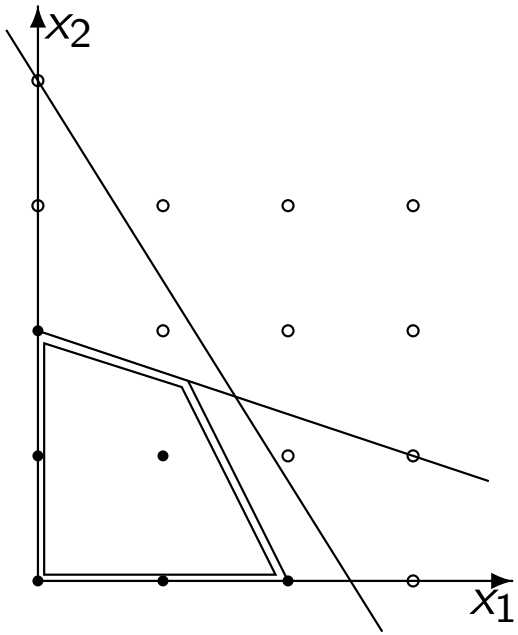
$$(\pi_1 A^1 - c)X_j + u = \left(0 * \boxed{\begin{matrix} 1 & 3 \end{matrix}} - \boxed{\begin{matrix} 1 & 1 \end{matrix}} \right) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 0 =$$

$$= -1\ x_1 - 1\ x_2$$

and Subproblem 1 is equivalent to :

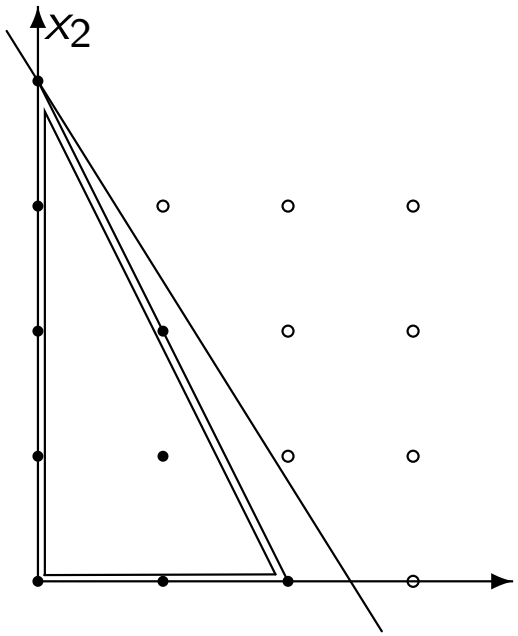
$$\begin{array}{ll} \max & x_1 + x_2 \\ \text{s. to} & x \in Conv\{x \in X \text{ and integer}\} \end{array}$$

Solution 1 in the space of original variables



To which solution in the space of the original variables x_1, x_2 does the current solution $\lambda_1 = 1, \lambda_2 = \lambda_3 = 0$ correspond ?

Graphical solution of subproblem 1



Which is the solution $x \in Conv\{x \in X \text{ and integer}\}$ that maximizes $x_1 + x_2$?

Insert column in restrict master problem and re-optimize

Solution of Subproblem 1

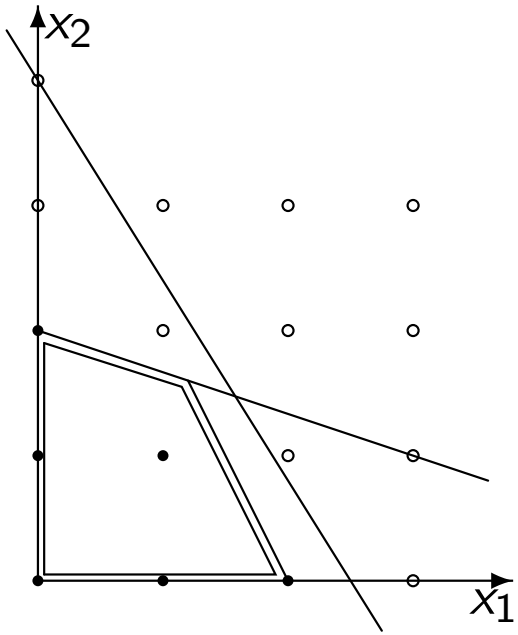
Optimal solution of subproblem 1 is $X_j^* = X_3 = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$ with objective solution value 4.

	z_{DW}	λ_3	s_1	λ_1	
s_1	0	12	1	0	6
λ_1	0	1	0	1	1
	1	-4	0	0	0

	z_{DW}	λ_3	s_1	λ_1	
λ_3	0	1	1/12	0	1/2
λ_1	0	0	-1/12	1	1/2
	1	0	1/3	0	2

Dual solution: $\pi_1 = 1/3, u = 0$

Solution 2 in the space of original variables



To which solution in the space of the original variables x_1, x_2 does the current solution $\lambda_1 = \lambda_3 = 1/2, \lambda_2 = 0$ correspond ?

Second subproblem:

Reduced cost:

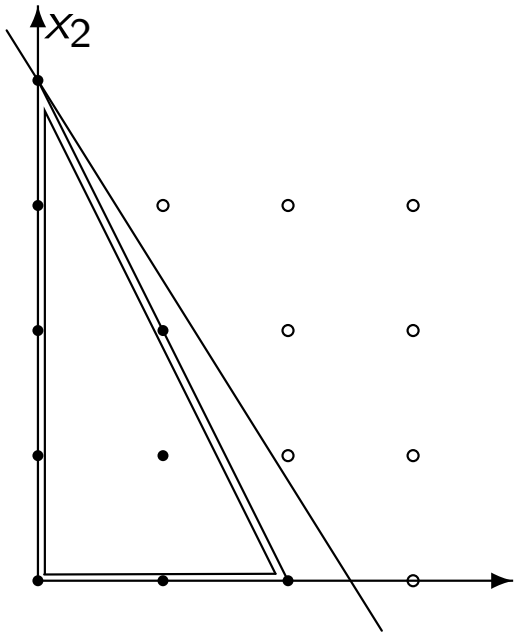
$$(\pi_1 A^1 - c)X_j + u = \left(\quad 1/3 \quad * \boxed{\begin{array}{cc} 1 & 3 \end{array}} - \boxed{\begin{array}{cc} 1 & 1 \end{array}} \right) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 0 =$$

$$= \quad -2/3\ x_1 - 0\ x_2$$

and Subproblem 2 is equivalent to :

$$\begin{array}{ll} \max & 2/3x_1 + 0x_2 \\ \text{s. to} & x \in Conv\{x \in X \text{ and integer}\} \end{array}$$

Graphical solution of subproblem 2



Which is the solution $x \in Conv\{x \in X \text{ and integer}\}$ that maximizes $2/3x_1 + 0x_2$?

Re-expressing the column

Optimal solution of subproblem 2 is $X_j^* = X_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ with objective solution value $4/3$.

To insert the new column $A_{X_2} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ in the simplex tableau, we need to express it in terms of the current base:

$$B^{-1}A_{X_2} = \begin{bmatrix} 1/12 & 0 \\ -1/12 & 1 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/6 \\ 5/6 \end{bmatrix}$$

$$c_B B^{-1}A_{X_2} - c_{X_2} = \begin{bmatrix} 1/3 & 0 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \end{bmatrix} - 2 = -4/3$$

This step was skipped in the last iteration, because $B^{-1} = I$.

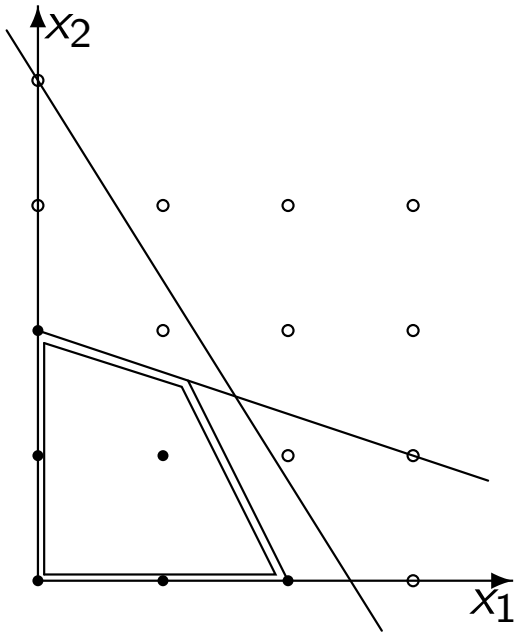
Second iteration

	z_{DW}	λ_2	λ_3	s_1	λ_1	
λ_3	0	1/6	1	1/12	0	1/2
λ_1	0	5/6	0	-1/12	1	1/2
	1	-4/3	0	1/3	0	2

	z_{DW}	λ_2	λ_3	s_1	λ_1	
λ_3	0	0	1	1/10	-1/5	2/5
λ_2	0	1	0	-1/10	6/5	3/5
	1	0	0	1/5	8/5	14/5

Dual solution: $\pi_1 = 1/5, u = 8/5$

Solution 3 in the space of original variables



To which solution in the space of the original variables x_1, x_2 does the current solution $\lambda_2 = 3/5,\ \lambda_3 = 2/5,\ \lambda_1 = 0$ correspond ?

Subproblem 3

Reduced cost:

$$(\pi_1 A^1 - c)X_j + u = \left(\quad 1/5 \quad * \boxed{\begin{array}{cc} 1 & 3 \end{array}} - \boxed{\begin{array}{cc} 1 & 1 \end{array}} \right) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 8/5 =$$

$$= \quad -4/5\ x_1 - 2/5\ x_2 + 8/5$$

and Subproblem 3 is equivalent to :

$$\begin{array}{ll} \max & 4/5x_1 + 2/5x_2 - 8/5 \\ \text{s. to} & x \in Conv\{x \in X \text{ and integer}\} \end{array}$$

Solution of subproblem:

- Optimal solution has objective function value equal to 0.
- There are no attractive solutions.

Solution of reformulated problem is optimal.

Lagrangean Relaxation vs. DW decomposition

$$
\begin{aligned}
z_{PI} = \min \quad & cx \\
s.to \quad & A^1 x \geq b^1 \\
& A^2 x \geq b^2 \\
& x \geq 0 \text{ and integer}
\end{aligned}
$$

Strategy

- keep a set of constraints in the problem (can be the empty set)
- relax constraints in remaining set; move them to the objective function, adding a penalty when they are not obeyed.
- relaxed constraints are indirectly enforced choosing a suitable penalty.
- Problem $RL(\lambda)$ is a relaxation of PI.

Lagrangean problem

Lagrangean problem, $RL(\lambda)$:

$$\begin{aligned} z_{RL}(\lambda) = \min \quad & cx + \lambda(b^2 - A^2x) \\ s.to \quad & A^1x \geq b^1 \\ & x \geq 0 \text{ and integer} \end{aligned}$$

where λ is a vector of non-negative Lagrange multipliers ($\lambda \geq 0$).

If a constraint of the second group is not obeyed, $A^2x < b^2$, then $\lambda(b^2 - A^2x)$ takes a positive value, which is a penalty for the objective function.

Rule for building Lagrangean problem:

violating a constraint yields a penalty for the objective function, no matter what are the type of constraints ($\leq, \geq$) and of the objective function (max. or min.).

Relation between Minimization Problems PI and $RL(\lambda)$

Problem $RL(\lambda)$ is a relaxation of the original problem PI.

Theorem

The optimal value of the lagrangean problem, $z_{RL}(\lambda)$, is always less than or equal to the value of the integer programming minimization problem, z_{PI}, for every $\lambda \geq 0$.

Proof: Any valid solution for problem PI is also valid for problem $RL(\lambda)$, because it obeys the 2 sets of constraints.

For any valid point x of problem PI, the lagrangean function $cx + \lambda(b^2 - A^2x) \leq cx$, because $\lambda \geq 0$ and $b^2 - A^2x \leq 0$.

In particular, this also happens for the optimal point of the integer programming point. Thus, $z_{RL}(\lambda) \leq z_{PI}$. □

Optimality conditions

Theorem

If, given a vector λ, the optimal solution x^* of the lagrangean problem satisfies the following conditions:

- x^* *is an optimal solution of* $RL(\lambda)$,
- $A^2x^* \geq b^2$,
- $\lambda(b^2 - A^2x^*) = 0$,

then the solution x^* is optimal for the original problem PI.

Proof: Solution x^* obeys all constraints and $z_{RL}(\lambda^*) = \min cx^* + \lambda(b^2 - A^2x^*) = cx^*$, which guarantees that the solution is optimal for the original problem. □

The complementary slackness condition (third item) has to be fulfilled:
- A solution x obeying the relaxed constraints, $A^2x \geq b^2$, may not be optimal for the original problem PI if $\lambda(b^2 - A^2x) < 0$.

Lagrangean dual problem

- There is a value of λ^* for which the lagrangean problem takes the largest value:

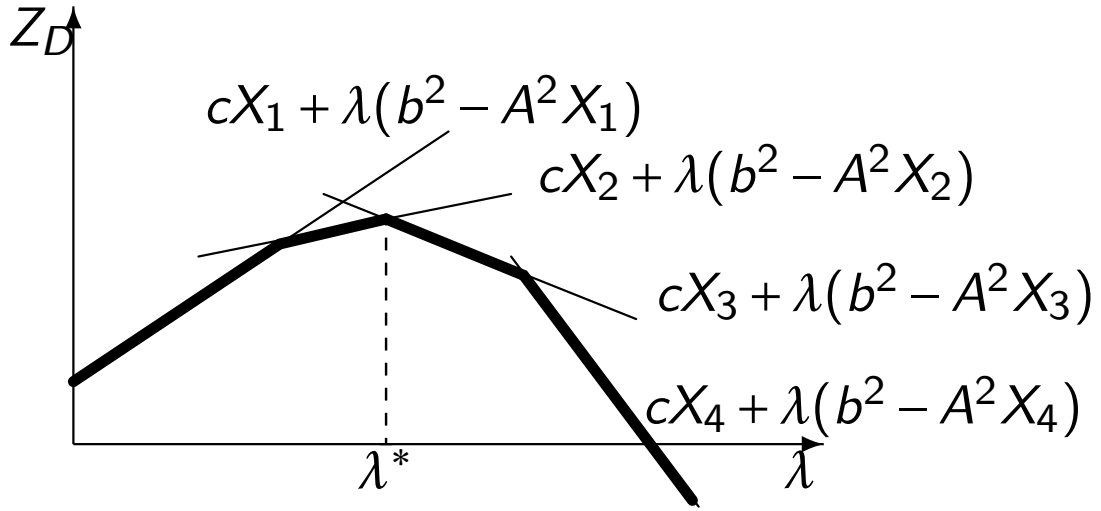
$$z_D = z_{RL}(\lambda^*) = \max_{\lambda} z_{RL}(\lambda).$$

- Problem D is denoted as the lagrangean dual problem:

$$\begin{aligned} z_D = \quad & \max_{\lambda}\{\min cx + \lambda(b^2 - A^2 x)\} \\ suj. \quad & A^1 x \geq b^1 \\ & x \geq 0 \text{ and integer} \end{aligned}$$

- usually solved with the subgradient method.

Lagrangean dual problem



- only the integer extreme points of $\{A^1x \geq b^1, x \geq 0 \text{ and integer}\}$ can be optimal solutions of the lagrangean dual problem.
- for each integer extreme point X_i, $cX_i + \lambda(b^2 - A^2X_i)$, $i = 1, 2, \ldots, I$, is a function of λ.
- function $\min cx + \lambda(b^2 - A^2x)$ is a concave piecewise linear function (shown in bold).

lagrangean dual problem: find the value of λ that maximizes the function $\min cx + \lambda(b^2 - A^2x)$

Lagrangean dual problem (another insight)

$$
\begin{aligned}
z_D = \quad & \max w \\
s.to \quad & w \leq cX_i + \lambda(b^2 - A^2 X_i), \ i = 1, 2, \ldots, I \\
& \lambda \geq 0 \\
& w \text{ unrestricted in sign}
\end{aligned}
$$

Decision variable w, for each value of λ, takes the value which is the smallest, among all integer extreme points, X_i, of the values of the function $cX_i + \lambda(b^2 - A^2 X_i)$.

Example

Please note that the constraints are swapped (for personal reasons).

$$
\begin{array}{lrrcrl}
\min & -1x_1 & -1x_2 & & & \\
s.to & -8x_1 & -5x_2 & \geq & -20 & (\text{ kept in problem}) \\
 & -1x_1 & -3x_2 & \geq & -6 & (\text{ relaxed}) \\
 & x_1, & x_2 \geq 0 \text{ and integer} & & &
\end{array}
$$

$c = (-1,-1)$, $A^1 = (-8,-5)$, $A^2 = (-1,-3)$, $b^1 = -20$ and $b^2 = -6$, for the extreme points, we get:

$$cX_1 = \begin{bmatrix} -1 & -1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0 \qquad A^2X_1 = \begin{bmatrix} -1 & -3 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$$cX_2 = \begin{bmatrix} -1 & -1 \end{bmatrix} * \begin{bmatrix} 2 \\ 0 \end{bmatrix} = -2 \qquad A^2X_2 = \begin{bmatrix} -1 & -3 \end{bmatrix} * \begin{bmatrix} 2 \\ 0 \end{bmatrix} = -2$$

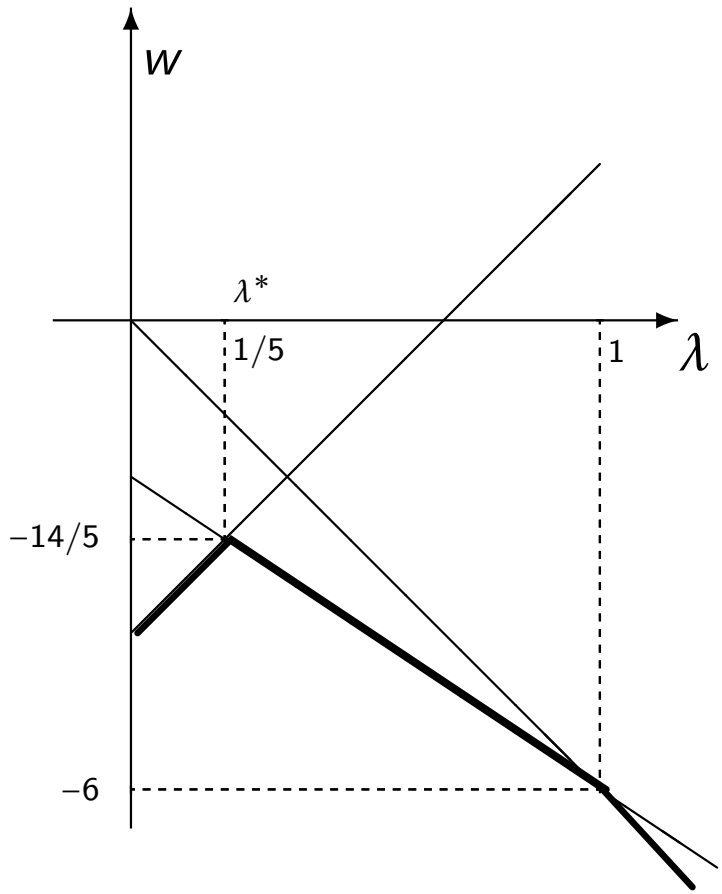
$$cX_3 = \begin{bmatrix} -1 & -1 \end{bmatrix} * \begin{bmatrix} 0 \\ 4 \end{bmatrix} = -4 \qquad A^2X_3 = \begin{bmatrix} -1 & -3 \end{bmatrix} * \begin{bmatrix} 0 \\ 4 \end{bmatrix} = -12$$

Example (cont.)

$$
\begin{aligned}
z_D = \quad & \max w \\
s.to \quad & w \leq cX_i + \lambda(b^2 - A^2 X_i), \ i = 1, 2, \ldots, I \\
& \lambda \geq 0 \\
& w \text{ unrestricted in sign}
\end{aligned}
$$

$$
\begin{aligned}
z_D = \quad & \max w \\
s.to \quad & w \leq 0 - 6\lambda \\
& w \leq -2 - 4\lambda \\
& w \leq -4 + 6\lambda \\
& \lambda \geq 0 \\
& w \text{ unrestricted in sign}
\end{aligned}
$$

Lagrangean dual function



The optimal solution is $w = -14/5$ and $\lambda = 1/5$, being the objective value of the optimal solution $-14/5$.

Building the dual of the Lagrangean Dual

Lagrangean dual has decision variables w and λ.

$$
\begin{aligned}
z_D = \quad & \max w \\
& w \leq cX_i + \lambda(b^2 - A^2 X_i), \ i = 1, 2, \ldots, I \\
& \lambda \geq 0 \\
& w \text{ unrestricted in sign}
\end{aligned}
$$

Dual variables: μ_i for each constraint i, $i = 1, 2, \ldots, I$,

- lagrangean dual has an exponential number of constraints.
- dual of lagrangean dual has an exponential number of decision variables.

Lagrangean dual - I

Lagrangean dual:

$$
\begin{aligned}
z_D = \quad & \max w \\
s.to \quad & w \leq cX_i + \lambda(b^2 - A^2 X_i),\ i = 1,2,\ldots,I \\
& \lambda \geq 0 \\
& w \text{ unrestricted in sign}
\end{aligned}
$$

Dual of Lagrangean dual:

$$
\begin{aligned}
z_D^d = \quad & \min \sum_{i=1}^{I} (cX_i)\mu_i \\
s.to \quad & \sum_{i=1}^{I} -(b^2 - A^2 X_i)\mu_i \geq 0 \\
& \sum_{i=1}^{I} \mu_i = 1 \\
& \mu_i \geq 0
\end{aligned}
$$

Lagrangean dual - II

As $\sum_{i=1}^{I} b^2 \mu_i = b^2 \sum_{i=1}^{I} \mu_i = b^2$, rewrite the problem as:

$$
\begin{aligned}
z_D^d = \quad & \min \sum_{i=1}^{I} (cX_i)\mu_i \\
s.to \quad & \sum_{i=1}^{I} (A^2 X_i)\mu_i \geq b^2 \\
& \sum_{i=1}^{I} \mu_i = 1 \\
& \mu_i \geq 0
\end{aligned}
$$

This model is the Dantzig-Wolfe decomposition that results from keeping in the master problem the set of constraints $A^2 x \geq b^2$ and putting in the subproblem the set of constraints $A^1 x \geq b^1$.

Example:

Lagrangean dual:

$$
\begin{aligned}
z_D = \quad & \max w \\
s.to \quad & w \leq 0 - 6\lambda \\
& w \leq -2 - 4\lambda \\
& w \leq -4 + 6\lambda \\
& \lambda \geq 0 \\
& w \text{ unrestricted in sign}
\end{aligned}
$$

Dual of Lagrangean dual:

$$
\begin{array}{lrrrcr}
\min & 0\mu_1 & -2\mu_2 & -4\mu_3 & & \\
s.to & 0\mu_1 & -2\mu_2 & -12\mu_3 & \geq & -6 \\
& \mu_1 & +\mu_2 & +\mu_3 & = & 1 \\
& \mu_1, & \mu_2, & \mu_3 \geq 0 & &
\end{array}
$$

Concluding remarks

- decomposition is a general tool to convert a difficult problem into a sequence of manageable problems.
- models from decomposition may be stronger.
- column generation has profited from developments in LP software.

Applications

Outline

- Reasons for using decomposition
- Block angular structure: examples
- Solving LP relaxations with column generation
- Application: Cutting Stock (CSP) and Bin Packing (BPP) Problems
- Application: Vehicle routing

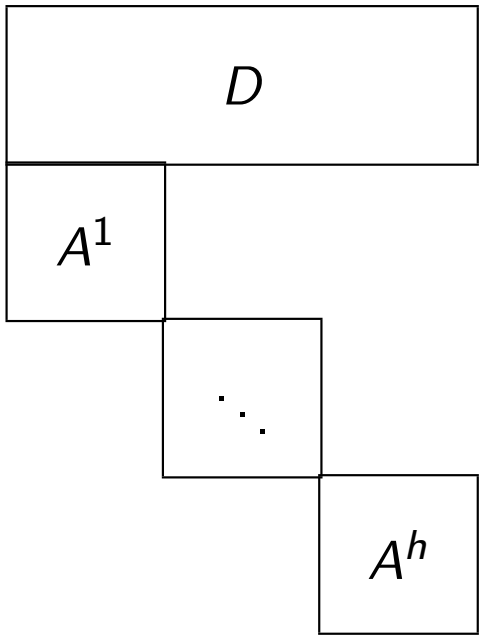
Reasons for using decomposition

Models from DW decomposition:

- become manageable in size: number of constraints is reduced and column generation is used.
- are suitable to deal with non-linear constraints: they are tackled in a dynamic programming subproblem.
- may be stronger: subproblems do not have the integrality property.
- may be the only models at hand, because compact models may not be known.

General structure: block angular with linking constraints

- DW decomposition partitions model into levels: Main problem and subproblem(s) (or Master and slave(s)).
- Subproblem(s) has(ve) nice structure that can be exploited (*e.g.*, network).



- Block D - Linking constraints
- Each of the blocks $A^1,\ldots,A^h$ defines a different subproblem

Examples of models resulting from structured problems

Problem	D block	A blocks
Production planning	Availability of common resources required for production (*e.g.*, machine capacities).	One block for each product. Production requirements of each product (for example, forced by existing demand).
Vehicle routing	Constraints imposed on the fleet of vehicles (*e.g.*, it must visit all the clients).	One for each vehicle. Route and vehicle constraints (*e.g.*, a route must end at a depot and vehicle capacity cannot be exceeded).
Generalised assignment	Constraints imposed on the group of agents (all the tasks must be performed).	One for each agent, related with its capacity.
Machine scheduling	Job constraints (*e.g.*, all jobs must be done).	One for each machine. Machine constraints (*e.g.*, two tasks cannot be made at the same time).

Decomposition framework

Master Problem (MP)

- "combines" independent solutions of SPs
- constraints in MP tell how resources are used by subproblem solutions

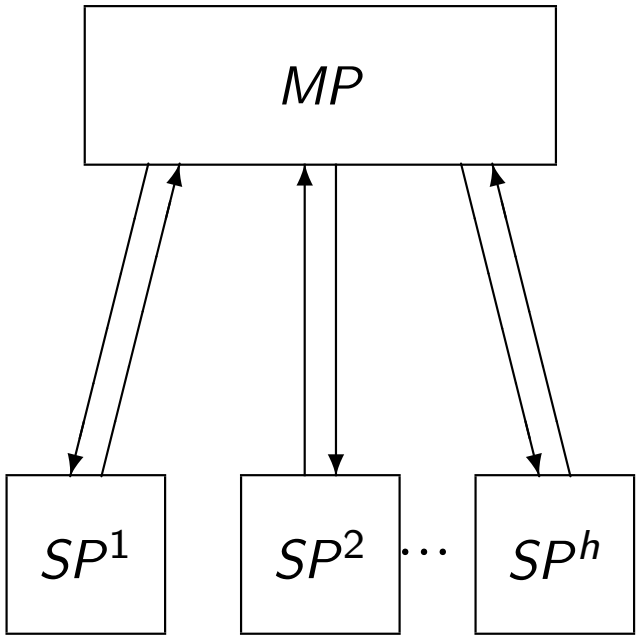
Subproblem(s) (SP)

- usually subproblem solutions are paths.
- difficult constraints (non-linearities) are tackled in the subproblem (solved with dynamic programming)
- SP use resources when economically efficient

Economic interpretation of DW decomposition

Master Problem (MP)

controls usage of resources: tells SP the price of the usage of resources



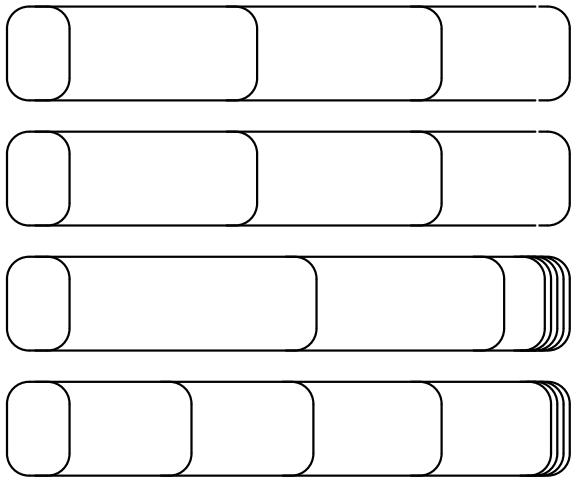
Subproblem(s) (SPs)

compete for resources: each SP makes its best bid to MP

Application I

- Cutting Stock Problem (CSP) and Bin Packing Problem (BPP)
- Kantorovich model
- Gilmore-Gomory model
- Solution of Gilmore-Gomory model by column generation
- Example (solution of LP relaxation)

Cutting Stock Problem



W: width of large rolls
w_i : width of rolls for client i, $i = \ldots, m$
b_i : demand of rolls of width w_i (many items of each size)
Objective: cut the minimum number of rolls to satisfy demand.

Cutting and packing problems

Bin packing problem:

- given an unbounded number of bins of capacity W and a list of n items of size w_i, $0 < w_i \leq W$, $i = 1, \ldots, n$,
- pack all the items in the smallest number of bins without exceeding their capacity.
- few items of each size.

Cutting stock problem:

- given an unbounded number of rolls of size W, and given m clients with demands of b_i rolls of size w_i, $0 < w_i \leq W$, $i = \ldots, m$,
- cut the minimum number of rolls to satisfy demand.
- many items of each size.

Cutting Stock Problem: a weak model

$$\text{Decision variables } x_{ij} = \begin{cases} 1 & \text{, if item } j \text{ is placed in roll } i \\ 0 & \text{, otherwise} \end{cases}$$

$$\text{Decision variables } y_i = \begin{cases} 1 & \text{, if roll } i \text{ is used} \\ 0 & \text{, otherwise} \end{cases}$$

$$\min z_{IP} = \sum_{i=1}^{n} y_i$$

$$\text{subj. to} \quad \sum_{j=1}^{n} w_j x_{ij} \leq W y_i, \ \forall i \in I$$

$$\sum_{i=1}^{n} x_{ij} = 1, \ \forall j \in J$$

$$y_i = 0 \text{ or } 1, \ \forall i$$

$$x_{ij} = 0 \text{ or } 1, \ \forall i,j$$

L. Kantorovich, "Mathematical methods of organising and planning production" (translated from a paper in Russian, dated 1939), Management Science 6, 366–422, 1960.

Quality of the relaxation: value of the lower bound

LP relaxation: replace the last two constraints by $0 \leq y_i \leq 1, \forall i$, and $0 \leq x_{ij} \leq 1, \forall i, j$, respectively.

LP relaxation optimum, z^*_{LP}, is a lower bound for the IP optimum.

Proposition (Martello and Toth, 90)

Lower bound $LB_1 = \lceil \sum_{i=1}^{n} w_i / W \rceil$.

Proof: No solution can have an objective value lower than $\sum_{i=1}^{n} w_i / W$.

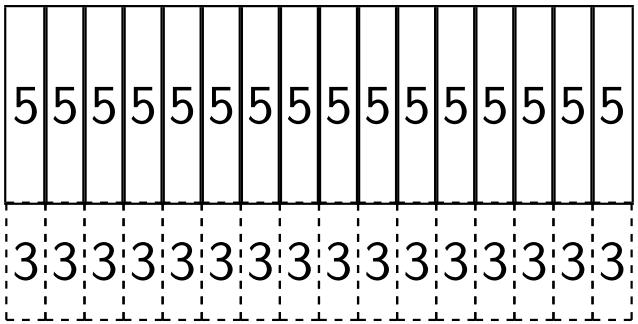
Solution $x_{ii} = 1, x_{ij} = 0, \forall j \neq i$, and $y_i = w_i / W, \forall i$, has an objective value, $z^*_{LP} = \sum_{i=1}^{n} w_i / W$, equal to that value. So, it is an optimal LP solution.

Round up, because the number of bins must be integer. □

Bound can be very poor for instances with large loss: there may be cases in which $z^*_{LP} \to 1/2\ z_{IP}$.

Example: Bins of capacity 8 and 16 items of size 5

Integer optimum is 16:



Linear relaxation optimum is 10:

$$\begin{array}{rcl} x_{11} = 1 & , & y_1 = 5/8 \\ x_{22} = 1 & , & y_2 = 5/8 \\ \ldots & & \ldots \\ x_{16,16} = 1 & , & y_{16} = 5/8 \end{array}$$

$$\sum_i y_i = 10$$

Gap between Integer and Linear Relaxation optima, $z_{IP} - z_{LP} = 6$.

Cutting Stock Problem: Gilmore-Gomory model

Cutting Pattern: possible arrangement of items in the roll:

$$\sum_{i=1}^{m} a_{ij} w_i \leq W$$
$$a_{ij} \geq 0 \text{ and integer, } \forall j \in J.$$

a_{ij} : number of items of width w_i obtained in the cutting pattern j
J : set of valid cutting patterns.
x_j : number of rolls cut according cutting pattern j.

$$\begin{aligned} min\ z_{IP} &= \sum_{j \in J} x_j \\ subj.\ to &\quad \sum_{j \in J} a_{ij} x_j \geq b_i,\ i = 1, 2, \ldots, m \\ &\quad x_j \geq 0 \text{ and integer, } \forall j \in J \end{aligned}$$

Example (cont.): Bins of capacity 8 and 16 items of size 5

The only valid cutting pattern is:

Mathematical formulation:

$$\begin{array}{lll} \min\ z_{LP} & = & x_1 \\ \text{subj. to} & & 1x_1 \geq 16 \\ & & x_1 \geq 0 \end{array}$$

Optimal value of linear relaxation $z_{LP} = 16$, when $x_1 = 16$.

Gap between Integer and Linear Relaxation optima, $z_{IP} - z_{LP} = 0$.

Gilmore-Gomory's bound in practice

- Bound given by the LP relaxation of Gilmore-Gomory's model is very tight.
- Most of the one-dimensional cutting stock instances have gaps smaller than one.
- There are instances with gaps equal to 1 (O.Marcotte'1985,86).
- Largest gap known is $\frac{7}{6}$ (Rietz,Scheithauer'2002).
- Conjecture: all instances have gaps smaller than 2 (modified integer round-up property) (Scheithauer,Terno'1995).

Column generation for CSP [Gilmore, Gomory, 1961]

Generally, it is unpractical to enumerate all valid cutting patterns.

Solve linear programming relaxation of CSP using column generation:

Choose an initial restricted set of cutting patterns
While (there is an a attractive cutting pattern) do
    add attractive cutting pattern to restricted problem
    reoptimize
End While

To get an integer solution, round up fractional values of cutting patterns.
Solutions are of good quality, if the quantities demanded are high.

Cutting Stock Problem: Restricted Problem

$$
\begin{aligned}
min \quad & z_{LP} = \sum_{j \in \bar{J}} x_j \\
subj.to \quad & \sum_{j \in \bar{J}} a_{ij} x_j \geq b_i, \ i = 1, 2, \ldots, m \\
& x_j \geq 0, \ \forall j \in \bar{J},
\end{aligned}
$$

$\bar{J}$: subset of cutting patterns in restricted problem
$\pi = \pi(\bar{J}) = (\pi_1, \pi_2, \ldots, \pi_m)$: optimal dual solution with subset $\bar{J}$

Pricing cutting patterns out of the restricted problem:

$$
\begin{aligned}
\text{Reduced cost of cutting pattern } j \quad &= \quad c_j - c_B B^{-1} A_j = \\
&= \quad 1 - \sum_{i=1}^{m} a_{ij} \pi_i
\end{aligned}
$$

Column is attractive if its reduced cost < 0

Cutting Stock Problem: objective function of subproblem

Find most attractive cutting pattern $\in J \setminus \overline{J}$:

$$\min_{j \in J \setminus \overline{J}} \; 1 - \sum_{i=1}^{m} a_{ij} \pi_i$$

Columns in $\overline{J}$ have reduced costs ≥ 0; so, search over J:

$$\min_{j \in J} \; 1 - \sum_{i=1}^{m} a_{ij} \pi_i$$

Maximize symmetric function:

$$\min_{j \in J} \; 1 - \sum_{i=1}^{m} a_{ij} \pi_i \;\equiv\; \max_{j \in J} \; \sum_{i=1}^{m} a_{ij} \pi_i - 1$$

Cutting Stock Problem: knapsack subproblem

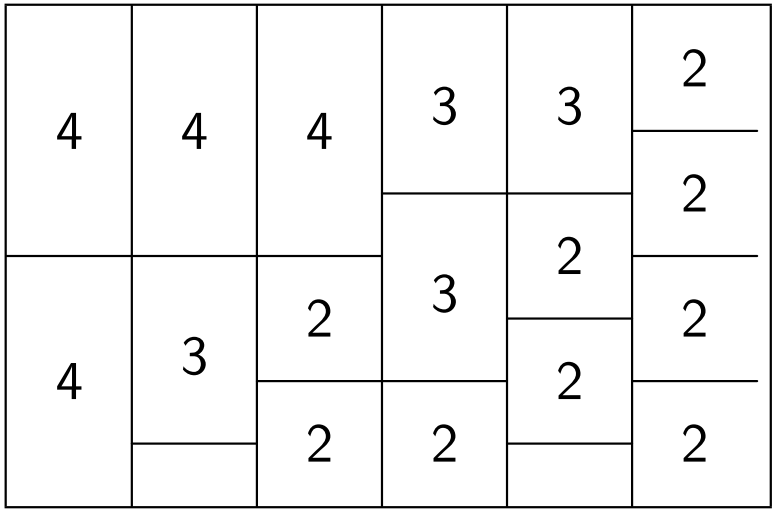
Knapsack subproblem:

$$\begin{aligned} \max\ z_s = \quad & \sum_{i=1}^{m} \pi_i y_i \\ \textit{subj. to} \quad & \sum_{i=1}^{m} w_i y_i \leq W \\ & y_i \geq 0 \text{ and integer, } i = 1, 2, \ldots, m, \end{aligned}$$

y_i : number of items of size w_i in the new cutting pattern

If optimum $z_s^* > 1$, cutting pattern is attractive.
If no attractive columns, solution is optimal.

(Very Small) Example



$W = 8$	cutting patterns						Demand b_i	
	x_1	x_2	x_3	x_4	x_5	x_6		
$w_i = 4$	2	1	1				$\geq$	5
3		1		2	1		$\geq$	4
2			2	1	2	4	$\geq$	8
min	1	1	1	1	1	1		

(Very Small) Example (cont.)

$W = 8$	cutting patterns						Demand b_i	
	x_1	x_2	x_3	x_4	x_5	x_6		
$w_i = 4$	2	1	1				$\geq$	5
3		1		2	1		$\geq$	4
2			2	1	2	4	$\geq$	8
min	1	1	1	1	1	1		

Optimal fractional solution

2.5	2.0	1.5

6 rolls

Fractional solution rounded up

3.0	2.0	2.0

7 rolls

Excess production: 1 item of width 4 and 2 items of width 2

Restricted problem: first iteration

Initial solution: 3 columns, each with items of the same size (as suggested by Gilmore and Gomory'61).

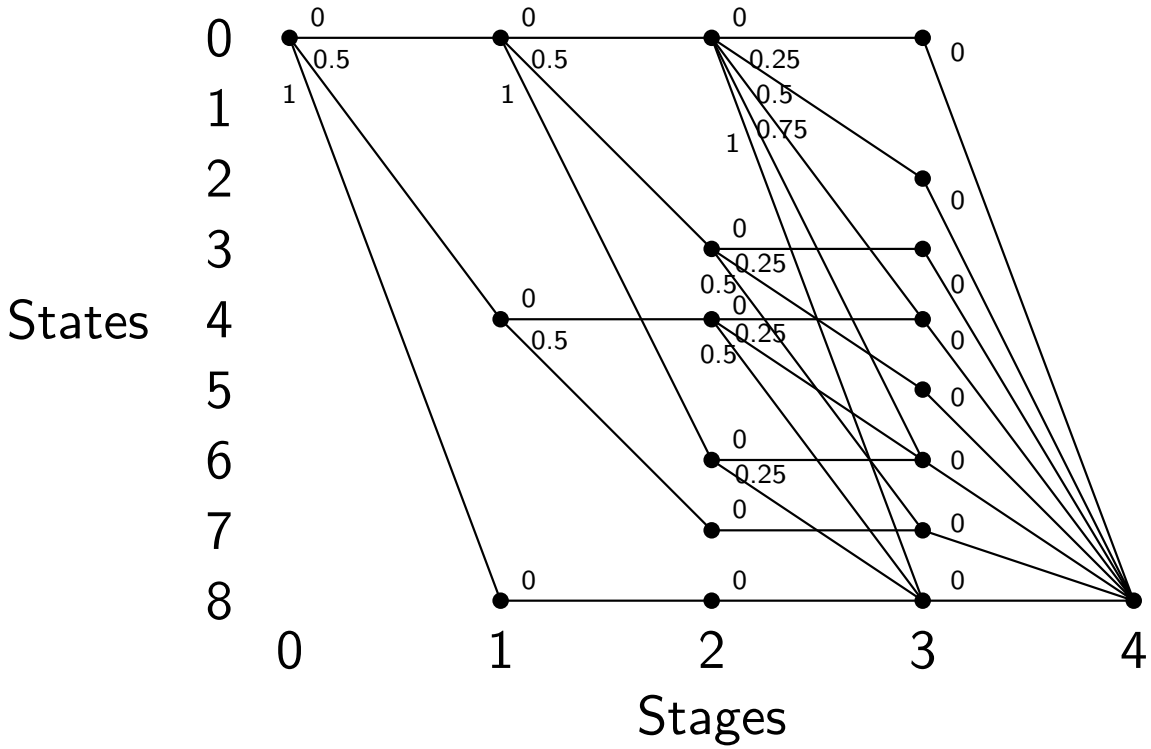
Using an LP solver, we obtain the following optimal solution (primal and dual):

	x_1	x_2	x_3			dual
$w_d =$ 4	2			$\geq$	5	0.5
3		2		$\geq$	4	0.5
2			4	$\geq$	8	0.25
min	1	1	1			

primal	2.5	2.0	2.0	$z^0 =$	6.5

Subproblem: first iteration

$$
\begin{array}{lrrrl}
\max\ z_s = & 0.5y_1 + & 0.5y_2 + & 0.25y_3 & \\
subj.\ to & 4y_1 + & 3y_2 + & 2y_3 & \leq 8 \\
 & y_j \geq 0 \text{ and integer}, \forall j & & &
\end{array}
$$



Optimal solution: $(y_1, y_2, y_3) = (0, 2, 1), z_s^* = 1.25 \rightarrow$ Attractive

Subproblem: knapsack problem

- Dynamic programming
- $F_i(x)$: maximum value from placing items with index less than or equal to i using x units of space.
- Recursive equations of Knapsack Problem with upper bounds on variables:

$$\begin{aligned} F_0(0) &= 0 \\ F_i(x) &= \max_{x - l w_i \geq 0} \{F_{i-1}(x - l w_i) + l\overline{\pi}_i : 0 \leq l \leq l_i^{max}\}, \end{aligned}$$

$$x = 0, 1, \ldots, W; \quad i = 1, 2, ..., m.$$

- Largest number of items of a given size in a cutting pattern (element a_{ij} in column j) must also be less than or equal to the demand of that size:

$$l_i^{max} = a_{ij}^{max} = \min\left\{b_i, \left\lfloor \frac{W}{w_i} \right\rfloor\right\}$$

- Computational complexity is $O(mW^2)$
- weakly NP-hard (pseudo-polynomial)

Restricted problem: second iteration

Attractive cutting pattern: 2 items of size 3 and 1 item of size 2.

Insert attractive column in the restricted problem, and reoptimize.

Optimal solution:

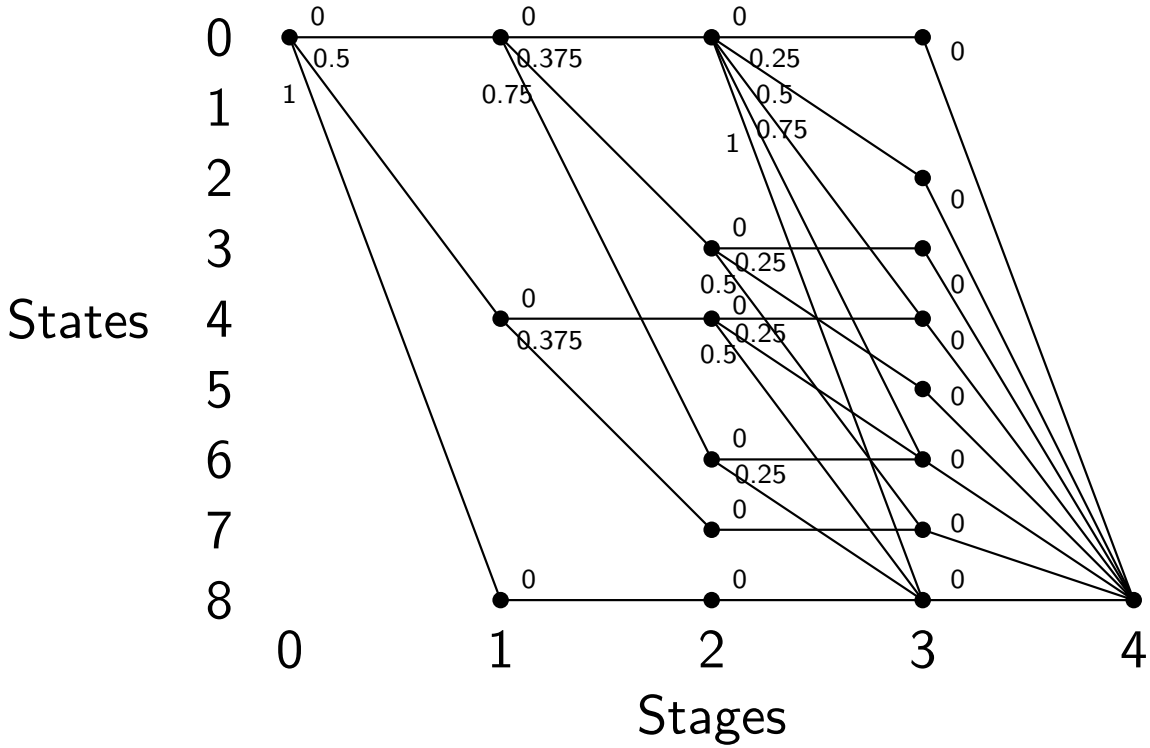
	x_1	x_2	x_3	x_4			dual
$w_d =$ 4	2				$\geq$	5	0.5
3		2		2	$\geq$	4	0.375
2			4	1	$\geq$	8	0.25
min	1	1	1	1			

primal	2.5	0.0	1.5	2.0

$z_{LP} =$ 6.0

Subproblem: second iteration

$$\begin{array}{lrrrl} \max\ z_s = & 0.5y_1 + & 0.375y_2 + & 0.25y_3 & \\ subj.\ to & 4y_1 + & 3y_2 + & 2y_3 & \leq 8 \\ & y_j \geq 0 \text{ and integer}, \forall j & & & \end{array}$$



Alternative optima (Value $z_s^* = 1.0$) → No attractive columns. So...

Optimal solution of the linear relaxation

	x_1	x_2	x_3	x_4			dual
$w_d =$ 4	2				$\geq$	5	0.5
3		2		2	$\geq$	4	0.375
2			4	1	$\geq$	8	0.25
min	1	1	1	1			

primal | 2.5 | 0.0 | 1.5 | 2.0 |

$z_{LP} =$ | 6.0 |

	A_1	A_3	A_4
4		2	3
		2	
4		2	3
		2	2
$x_j =$	2.5	1.5	2.0

Strengthening the formulation

LP relaxation has an optimal value z_{LP}. Optimal solution has an integer value.
Round-up: use a number of rolls $\geq$ LP optimum rounded up:

$$\sum_{j \in \bar{J}} x_j \quad \geq \quad \lceil z_{LP} \rceil$$

In this case, z_{LP} is integer: new constraint does not change the optimal solution.

	x_1	x_2	x_3	x_4			dual
$w_d =$ 4	2				$\geq$	5	0.5
3		2		2	$\geq$	4	0.375
2			4	1	$\geq$	8	0.25
round-up	1	1	1	1	$\geq$	6	0.0
min	1	1	1	1			

primal	2.5	0.0	1.5	2.0

$z_{LP} =$ 6.0

Easy to transfer dual information to subproblem.

Application II

- Vehicle Routing Problem with Time Windows
- Flow model
- Reformulated model
- Subproblem
- Dealing with subproblem
- Resource constraints: a general framework

Vehicle Routing Problem with Time Windows

Statement

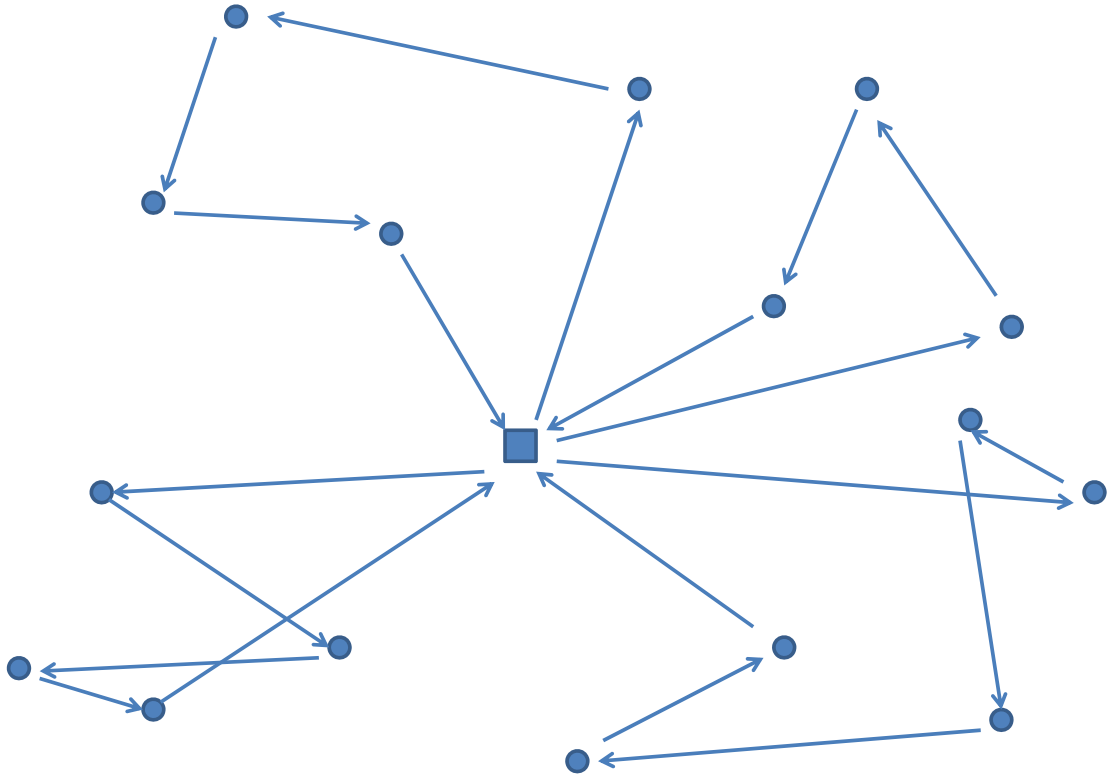
Given

- a set of vehicles with given capacities,
- a set clients with given demands and time windows,

find

- a set of routes, all starting and ending at the depot,
- such that each client is visited by one vehicle in a way that minimizes costs.

A set of routes



Mathematical model

Set of clients $N = \{1, 2, \ldots, n\}$

- demands d_i, $i \in N$
- time windows $[a_i, b_i]$, $i \in N$.

Set of homogeneous vehicles $\{1, 2, \ldots, K\}$

- K is known
- capacity Q

Mathematical model

Single depot, which is the origin and the destination of all vehicle routes:

split into 2 nodes:

- origin node $o \equiv$ vertex 0
- destination node $d \equiv$ vertex $n+1$
- no demand: $d_0 = d_{n+1} = 0$
- time windows $[a_0, b_0] = [a_{n+1}, b_{n+1}]$

Mathematical model

Graph $G = (V, A)$

- $V = N \cup \{o, d\}$ represents the set of nodes
- A the set of oriented arcs.

arc $(i, j) \in A \subset V \times V$:

- c_{ij} : cost incurred in travelling through the arc
- t_{ij} : travel time (includes service time of node i)
- for an arc to be feasible,

$$a_i + t_{ij} \leq b_j$$

The optimization objective of the plan is to minimize the total cost of the vehicles routes.

Mathematical model

Feasible route: path $p = (v_0, v_1, \ldots, v_H)$

- starts at origin node $(v_0 = o)$
- ends at destination node $(v_H = d)$
- visits customers $v_i \in N,\ i = 1, \ldots, H-1$
- obeys capacity constraints $\sum_{i=1}^{H-1} d_i \leq Q$
- obeys time windows:

$$\begin{aligned} T_0 &= a_{v_0} \\ T_{i+1} &= \max\{a_{v_i+1}, T_i + t_{v_i,v_{i+1}}\} \leq b_{v_{i+1}},\ \forall i = 0, \ldots, H-1 \end{aligned}$$

Decision variables

Flow variables:

- $x_{ij}^k = \begin{cases} 1 & \text{, if vehicle } k \text{ travels from client } i \text{ to client } j \\ 0 & \text{, otherwise} \end{cases}$

  $\forall k = 1, \ldots, K, (i,j) \in A$

Time variables:

- T_i^k : start of service of vehicle k at node i

  $\forall k = 1, \ldots, K, i \in V$

Model with arc variables

$$\min \quad \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{1}$$

$$s.to \quad \sum_{k=1}^{K} \sum_{(i,j)\in\delta^+(i)} x_{ij}^k = 1, \ \forall i \in N \tag{2}$$

$$\sum_{(0,j)\in\delta^+(0)} x_{0j}^k = 1, \ \forall k = 1,\ldots,K \tag{3}$$

$$\sum_{(i,j)\in A} d_i x_{ij}^k \leq Q, \ \forall k = 1,\ldots,K \tag{4}$$

$$\sum_{(i,j)\in\delta^-(j)} x_{ij}^k = \sum_{(j,i)\in\delta^+(j)} x_{ji}^k, \ \forall j \in N, k = 1,\ldots,K \tag{5}$$

$$\sum_{(i,d)\in\delta^-(d)} x_{id}^k = 1, \ \forall k = 1,\ldots,K \tag{6}$$

$$T_i^k - T_j^k + M x_{ij}^k \leq M - t_{ij}, \ \forall k = 1,\ldots,K, (i,j) \in A \tag{7}$$

$$a_i \leq T_i^k \leq b_i, \ \forall k = 1,\ldots,K, i \in V \tag{8}$$

$$x_{ij}^k \in \{0,1\}, \ \forall k = 1,\ldots,K, (i,j) \in A \tag{9}$$

Time constraints (7)

$$T_i^k - T_j^k + Mx_{ij}^k \leq M - t_{ij},\ \forall k = 1,\ldots,K,(i,j) \in A$$

- $M = b_i - a_j + t_{ij}$ provides a tighter constraint
- an alternative way of expressing constraint is the *non-linear* constraint:

$$(T_i^k - T_j^k + t_{ij})x_{ij}^k \leq 0,\ \forall k = 1,\ldots,K,(i,j) \in A$$

Dantzig-Wolfe decomposition

- Keep in the master problem the partitioning constraints
- Remaining constraints in subproblem k
- Subproblem k finds solutions which are *elementary shortest paths* with capacity constraints and time windows
- extreme points are feasible route $\equiv$ paths
- each decision variable corresponds to a path for vehicle k
- if the fleet is homogeneous, all blocks are identical

Reformulated model

- P^k : set of feasible routes for vehicle k, each obeying the constraints,
- $y_p^k \in \{0,1\}$: vehicle k does route $p \in P^k$
- $c_p^k = \sum_{i=0}^{h} c_{v_i,v_{i+1}}^k$: cost of vehicle k in path $p \in P^k$

$$\begin{aligned}
\text{min} \quad & \sum_{k=1}^{K} \sum_{p \in P^k} c_p^k y_p^k \\
\text{s. to} \quad & \sum_{k=1}^{K} \sum_{p \in P^k} \delta_{ip}^k y_p^k = 1 \ , \ \forall i \in V \\
& \sum_{p \in P^k} y_p^k = 1, \ k = 1, \ldots, K \\
& y_p^k \in \{0,1\}, \ \forall p \in P^k, k = 1, \ldots, K
\end{aligned}$$

- where

$$\delta_{ip}^k = \begin{cases} 1 & \text{, if route } p \text{ of vehicle } k \text{ visits client } i \\ 0 & \text{, otherwise} \end{cases}$$

Reformulated model with all vehicle identical

- $P = P^k,\ k = 1,\ldots,K$: all vehicles are identical
- convexity constraints $\sum_{p \in P^k} y_p^k = 1,\ k = 1,\ldots,K$ can be aggregated into a single constraint $\sum_{k=1}^{K} \sum_{p \in P^k} y_p^k = K$.
- Vehicles with path (o,d) do not leave the depot;
- they act as slack variables: above constraint is a $\leq$ constraint.
- Using $y_p = \sum_{k=1}^{K} y_p^k$, then

$$
\begin{aligned}
\min \quad & \sum_{p \in P} c_p y_p \\
\text{s. to} \quad & \sum_{p \in P} \delta_{ip} y_p = 1\ ,\ \forall i \in V \\
& \sum_{p \in P} y_p \leq K \\
& y_p \in \{0,1\},\ \forall p \in P
\end{aligned}
$$

where

$$
\delta_{ip} = \begin{cases} 1 & \text{, if route } p \text{ visits client } i \\ 0 & \text{, otherwise} \end{cases}
$$

Example with 8 clients

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	
node 1	1	1	1	1	1	1										$= 1$
2	1	1					1	1	1	1	1					$= 1$
3			1	1			1					1	1	1		$= 1$
4	1							1	1			1				$= 1$
5					1					1					1	$= 1$
6		1	1					1					1			$= 1$
7				1			1				1	1		1		$= 1$
8						1					1		1	1	1	$= 1$
vehicles	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	≤ 4
cost	8	7	10	9	8	7	10	11	7	6	10	9	10	12	7	

Subproblem

Find path with minimum reduced cost:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \overline{c}_{ij} x_{ij} \\
s.to \quad & \sum_{(0,j)\in \delta^+(0)} x_{0j} = 1 \\
& \sum_{(i,j)\in A} d_i x_{ij} \leq Q \\
& \sum_{(i,j)\in \delta^-(j)} x_{ij} = \sum_{(j,i)\in \delta^+(j)} x_{ji}, \ \forall j \in N \\
& (T_i - T_j + t_{ij}) x_{ij} \leq 0, \ \forall (i,j) \in A \\
& a_i \leq T_i \leq b_i, \ \forall i \in V \\
& x_{ij} \in \{0,1\}, \ \forall (i,j) \in A
\end{aligned}
$$

Subproblem

Elementary Shortest Path Problem with Time Windows and Capacity constraints (ESPPTWC)

- path p
- starts at origin node o, and ends at destination node $n+1$
- obeys capacity constraints
- obeys time windows, and
- with minimum reduced cost $\overline{c}_p = \sum \overline{c}_{ij} x_{ij}$

Reduced costs of arcs:

- $\overline{c}_{ij} = c_{ij} - \pi_i, \ \forall (i,j) \in A, i \neq o$
- $\overline{c}_{ij} = c_{ij} - \mu, \ \forall (o,j) \in A$, where
- π_i : dual variable of visit constraint to node i,
- μ : dual variable of number of vehicles constraint

arcs with negative reduced cost induce negative cost cycles in the network!

ESPPTWC

- Dynamic programming
- $F(S,i,t)$: minimum cost of path from o to $i, i \in N \cup \{d\}$, visiting all nodes in set $S \subseteq N \cup \{d\}$ *only once,* and servicing node i at time t or later.
- Recursive equations:

$$\begin{aligned} F(\emptyset, o, a_0) &= 0 \\ F(S,j,t) &= \min_{(i,j)\in A}\{F(S-\{j\}, i, t') + c_{ij} : \\ & \quad i \in S-\{j\}, t' \leq t - t_{ij}, a_i \leq t' \leq b_i\}, \\ & \quad \forall S \subseteq N \cup \{d\}, j \in S, a_j \leq t \leq b_j \end{aligned}$$

- Optimal solution:

$$\min_{S\subseteq N\cup\{d\}} \min_{a_d \leq t \leq b_d} F(S,d,t)$$

- Strongly NP-hard (and very hard to solve in practice)

Relax elementary path requirement to get an easier problem:

Shortest Path Problem with Time Windows and Capacity constraints

- $F(i,t)$: minimum cost of path from o to $i, i \in N \cup \{d\}$, and servicing node i at time t or later.
- Recursive equations:

$$\begin{aligned} F(o,a_0) &= 0 \\ F(j,t) &= \min_{(i,j)\in A}\{F(i,t') + c_{ij} : \\ & \quad i \in N \cup \{d\} - \{j\}, t' \leq t - t_{ij}, a_i \leq t' \leq b_i\}, \\ & \quad \forall j \in N \cup \{d\}, a_j \leq t \leq b_j \end{aligned}$$

- Optimal solution:

$$\min_{a_d \leq t \leq b_d} F(d,t)$$

- now, node i can be visited more than once in path p,
- in the RMP, we get coefficients δ_{ip} that may be larger than 1.
- Weakly NP-hard (pseudo-polynomial)

Different relaxations of ESPPTWC

It is possible to design a dynamic programming recursion that eliminates some of the cycles generated in the solution of the subproblem.

- SPPTWC: all cycles are allowed
- SPPTWC-2-cycles: 2-cycles are not allowed
- SPPTWC-k-cycles: cycles of length $\leq k$ are not allowed

Trade-off:

- larger values of k produce stronger LP-relaxation of master problem
- larger values of k induce much more complex recursion, more difficult to code

Shortest path problem with resource constraints (SPPRC)

Desrochers'86:

Generalization when there is a set of resources

Set of resources R

- travel time t_{ij} is replaced by the consumption of t_{ij}^r units of resource $r, \forall r \in R$
- time interval constraint $[a_i, b_i]$ of node i is replaced by $|R|$ constraints $[a_i^r, b_i^r]$, $\forall r \in R$
- T_i^r : amount of resource r used to reach node i, starting from o
- a path using less than a_i^r resources to reach node i wastes some resources and is feasible
- a path using more than b_i^r resources to reach node i is infeasible
- For an arc (i,j) to be feasible

$$a_i^r + t_{ij}^r \leq b_j^r, \ \forall r \in R$$

SPPRC

- extension of classical shortest path problem
- cost is replaced by multidimensional resource vector
- resources are accumulated / propagated along arcs
- resource values are constrained at the nodes
- Additional material

S. Irnich and D. Villeneuve, The Shortest-Path Problem with Resource Constraints and k-Cycle Elimination for $k \geq 3$, INFORMS Journal on Computing 18(3), pp. 391.406, 2006.

Concluding remarks

- column generation is an intuitive framework.
- very effective in many applications.
- field is growing.

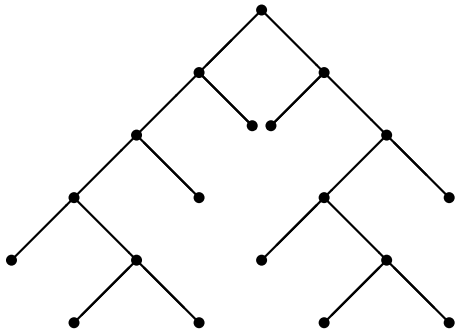
Branch-and-price algorithms

Outline

- Partition and branching
- Compatibility between master and sub-problem
- Coping with changes in sub-problem
- Application (binary variables): parallel machine scheduling

Getting integer solutions with branch-and-price

Branch-and-price = branch-and-bound + column generation



Methodology

- Branching constraints are introduced in the restricted master.
- After branching, deep in the tree, new columns may be needed.
- Column generation still has to work correctly.

Compatibility between Master Problem and Subproblem

Structure of the restricted master problem

- Branching constraints change the structure of the restricted master problem.
- Subproblem has to identify correctly the attractive and non-attractive columns with respect to the new structure.

Compatible (or *robust*) branching scheme

- Desirably, subproblem should be the same optimization problem both during the linear relaxation and branch-and-price.

Coping with changes

- Branching scheme should not induce intractable changes in the structure of the subproblem.

Branching schemes

Branching on variables of the reformulated model

- Regeneration of variables: a column set to zero by a branching constraint in the restricted master problem may turn out to be the most attractive column generated by the subproblem.

Branching on original variables

- Original variables: variables of model to which the Dantzig-Wolfe decomposition is applied.
- Successful in many applications.
- Often, original variables are related with flows in arcs.

A review of Partition and Branching

Each node of a branch-and-bound tree corresponds to a set of solutions obeying the constraints of the original problem and all branching constraints down to the node.

Partition:

- divides set of solutions into (desirably) mutually exclusive subsets,
- (should be a polynomial number of subsets),
- (desirably) corresponding to problems of the same type,
- eliminating the fractional solution of the node.

Basic and balanced branching schemes

- Basic branching rule: pick fractional x_{ij} and create 2 branches:

$$x_{ij} \leq \lfloor x_{ij} \rfloor$$
$$\text{and}$$
$$x_{ij} \geq \lceil x_{ij} \rceil$$

- Acting on a single variable may lead to a dive in the branching tree where no solutions will be found, and we still have to explore the other branch.
- Usually, better branching rules can be devised leading to more *balanced trees*, where we expect the subtrees to be of similar size.
- Branching schemes with unbalanced trees may perform very well on some instances, but very poorly on others.

Balanced branching schemes

Example 1: $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 1$

- Given the fractional solution $x_1 = 2/3$ and $x_6 = 1/3$,
- instead of using pair of branching constraints $x_1 \geq 1$ and $x_1 \leq 0$,
- use $x_1 + x_2 + x_3 \geq 1$ and $x_1 + x_2 + x_3 \leq 0$

Example 2: problems based on flows on arcs:

- For each node i, compute outflows $\sum_{j \in J} x_{ij}$
- Select set of successors $\overline{J}$: $\sum_{j \in \overline{J}} x_{ij}$ has a fractional value α,
- Use branching constraints:
  - $\sum_{j \in \overline{J}} x_{ij} \geq \lceil \alpha \rceil$
  - $\sum_{j \in \overline{J}} x_{ij} \leq \lfloor \alpha \rfloor$

Example

- left branch: $x_{12} + x_{13} \geq 2$
- right branch: $x_{12} + x_{13} \leq 1$

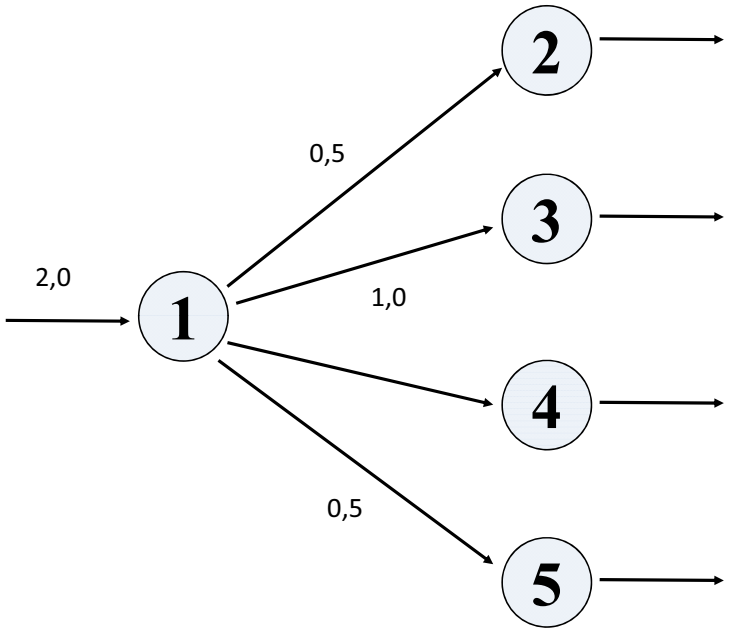


Figure shows part of a network

Selection of branching constraint

Branch first on decisions that have a larger impact on the solution.

Example 1: variable size bin-packing problem

- use a two-level branching scheme:
- branch first on bins until a bin integer solution is found, and then branch on items.
- in both levels, start with larger pieces (bins and items).

Branching in branch-and-price

Binary variables

- Ryan and Foster's rule for set partitioning problems
- Application: GAP
- Application: BCSP
- Application: multicommodity flow problems
- Application: vehicle routing

General integer variables

- general strategy
- adding branching constraints to the master problem explicitly
- Application: Cutting Stock Problems

Set partitioning problems

- S : finite set with m elements,
- $S_1, S_2, \ldots, S_n$, a collection of subsets of S.
- A partition of S is a collection of subsets, $S_{i_1}, \ldots, S_{i_j}, \ldots, S_{i_K}$, identified by $i_1, \ldots, i_j, \ldots, i_K$, such that:

$$\begin{aligned} \cup_{j=1}^{k} S_{i_j} &= S \\ S_{i_i} \cap S_{i_j} &= \emptyset, \ \forall i, j \end{aligned}$$

Ryan and Foster's rule (1981) for set partitioning problems

- Branches on variables of set partitioning problem:
- $\min\{cx : Ax = 1, x \in \{0,1\}^n\}$, and $A \in \{0,1\}^{m \times n}$.
- In the optimal integer solution of a set partitioning problem, each row is covered by ⋆exactly⋆ one column (variable).

Proposition

Given a fractional solution to $Ax = 1, x \geq 0$, *there are rows* r *and* s *such that* $0 < \sum_{j:a_{rj}=a_{sj}=1} x_j < 1$.

Many reformulated (column generation) models are set partitioning problems.

Branching rule

	x_{j_1}	x_{j_2}		
r	1	1	=	1
s	1		=	1

Branching rule: create two branches

- in left branch, rows r and s are covered by the same columns, *i.e.*, $\sum_{j:a_{rj}=a_{rs}=1} x_j = 1$.
- in right branch: rows r and s are covered by different columns, *i.e.*, $\sum_{j:a_{rj}=a_{rs}=1} x_j = 0$.

Example

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	1	1		1				$1 = 1$
2		1	1		1			$1 = 1$
3	1		1			1		$1 = 1$
	0.5	0.5	0.5					

For rows 1 and 2:

- left branch: $x_2 + x_7 = 1$
- right branch: $x_2 + x_7 = 0$

Dealing with branching constraints

	x_{j_1}	x_{j_2}		
r	1	1	=	1
s	1		=	1

A column j is feasible in the master problem,

- in left branch: if $(a_{rj} = a_{sj} = 1)$ or $(a_{rj} = a_{sj} = 0)$
- in right branch: if $(a_{rj} = a_{sj} = 0)$ or $(a_{rj} = 1, a_{sj} = 0)$ or $(a_{rj} = 0, a_{sj} = 1)$.

In the subproblem,

- in left branch: only accept solutions in which row 1 and 2 are both covered
- in right branch: if solution covers one row, the other must not be covered

Example: generalized assignment problem (GAP)

- maximize profit of assigning a set of jobs to agents with capacities $W_i, \forall i$.
- job j uses w_{ij} units of capacity of agent i.

$$\text{assignment variables } x_{ij} = \begin{cases} 1 & \text{, if job } j \text{ is assigned to agent } i \\ 0 & \text{, otherwise} \end{cases}$$

$$\begin{aligned} \max z &= \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \\ \text{subj. to} \quad & \sum_{j=1}^{n} w_{ij} x_{ij} \leq W_i, \ i = 1, \ldots, m \\ & \sum_{i=1}^{m} x_{ij} = 1, \ j = 1, \ldots, n \\ & x_{ij} \in \{0, 1\}, \ \forall i, j \end{aligned}$$

- Applications: in vehicle routing, resource scheduling, ...

Reformulated (set partitioning) model

- $K_i = \{x_1^i, x_2^i, \ldots, x_{k_i}^i\}$: set of all feasible assignments to agent i, $\forall i$.
- feasible assignment is a 0,1 vector $x_k^i = (x_{1k}^i, x_{2k}^i, \ldots, x_{nk}^i)$.

$$\text{ref. variables: } y_k^i = \begin{cases} 1, & \text{if feasible assignment } x_k^i \text{ is used for agent } i \\ 0, & \text{otherwise} \end{cases}$$

$$\forall i = 1, \ldots, m, \ k \in K_i.$$

$$\max z = \sum_{i=1,\ldots,m,\ k \in K_i} (\sum_{j=1}^{n} p_{ij} x_{jk}^i) y_k^i$$

$$\text{subj. to} \quad \sum_{i=1,\ldots,m,\ k \in K_i} x_{jk}^i y_k^i = 1, \ j = 1, \ldots, n$$

$$\sum_{k \in K_i} y_k^i \leq 1, \ i = 1, \ldots, m$$

$$y_k^i \in \{0,1\}, \ i = 1, \ldots, m, \ k \in K_i$$

- m knapsack subproblems, one for each agent.

Example with 5 jobs and 3 agents

	y_1^1	y_2^1	y_3^1	y_4^1	y_5^1	y_6^1	y_7^1	y_1^2	y_2^2	y_3^2	y_4^2	y_1^3	y_2^3	y_3^3	
job 1	1	1	1									1			$=1$
2	1			1				1	1			1			$=1$
3		1			1			1		1			1		$=1$
4			1		1	1			1	1	1		1	1	$=1$
5				1		1	1				1			1	$=1$
agent 1	1	1	1	1	1	1	1								≤ 1
2								1	1	1	1				≤ 1
3												1	1	1	≤ 1
max	7	8	6	5	4	9	5	6	4	6	4	5	3	5	

- set partitioning problem: instead of $\leq$ constraints, use "idle agent" columns (*i.e.*, slack variables for agent constraints).
- agent constraints are convexity constraints.

Branching rule [Savelsbergh'97]

Ryan and Foster's rule with one row belonging to a job and another row belonging to an agent $\Rightarrow$ branch on original variables x_{ij}.

Branching rule: create two branches

- in left branch $(x_{ij} = 1)$: agent i does job j
- in right branch $(x_{ij} = 0)$: assign job j to an agent i' other than i .

Branching constraints are not added explicitly to the Restricted Master Problem, but actions are taken to guarantee that the solution of a given node obeys the branching constraints imposed on the node.

Branching rule [Savelsbergh'97] (cont.)

left branch $(x_{ij} = 1)$:

- in master problem, set to 0:
  - all columns y_k^i of agent i that do not make job j (with $x_{jk}^i = 0$)
  - all columns $y_k^{i'}$ of agents i' other than i that make job j (with $x_{jk}^{i'} = 1$)
- in subproblem: always include job j in knapsack solution of agent i:

$$\begin{aligned} \max\ z = \quad & (\sum_{s \in S} \pi_s y_s) + \pi_j \\ subj.to \quad & \sum_{s \in S} w_s y_s \leq W_i - w_j \\ & y_s \in \{0,1\},\ \forall s \in S = \{1,\ldots,n\} \setminus \{j\} \end{aligned}$$

right branch $(x_{ij} = 0)$:

- in master problem: set to 0 all columns y_k^i of agent i that make job j
- in subproblem: exclude job j from knapsack subproblem of agent i.

Example: binary cutting stock problem (BCSP)

Binary cutting stock problem:

- demand constraints are disaggregated, and items are treated separately.
- only practical when quantities demanded by each client are very small, close to 1 unit.

$$
\begin{aligned}
min\ z_{IP} \quad &= \quad \sum_{j \in J} x_j \\
subj.\ to \quad & \quad \sum_{j \in J} a_{ij} x_j = 1,\ i = 1, 2, \ldots, m \\
& \quad x_j \in \{0, 1\},\ \forall j \in J
\end{aligned}
$$

- Special case of GAP when all agents are identical.

Example with 5 items

$W = 8$	cutting patterns						Demand b_i	
	x_1	x_2	x_3	x_4	x_5	x_6		
$w_i = 4$	1	1	1				$=$	1
4	1			1	1		$=$	1
3		1		1		1	$=$	1
2			1		1	1	$=$	1
2			1		1	1	$=$	1
min	1	1	1	1	1	1		

(some patterns may be missing...)

Set partitioning model:

- there are no convexity constraints (bins are equal).
- (convexity constraints would appear if bins were treated separately: same set of feasible solutions for every bin).
- no so structured as GAP.

Branching rule [Vance *et al.* '94]

Constraints are not added explicitly to the Master Problem.

in left branch: items r and s must belong to the same bin

- in Master Problem: combine rows r and s in a single row, and set to 0 all columns that are infeasible,
- in (knapsack) Subproblem: replace items r and s by a single item $m+1$ with $\pi_{m+1} = \pi_r + \pi_s$ and $w_{m+1} = w_r + w_s$.

$$\begin{aligned} \max\ z_s = \quad & \sum_{i \in S} \pi_i y_i \\ \textit{subj. to} \quad & \sum_{i \in S} w_i y_i \leq W \\ & y_i \in \{0,1\},\ \forall i \in S = \{1,\ldots,m\} \backslash \{r,s\} \cup \{m+1\} \end{aligned}$$

- subproblem generates columns with either both items r and s or none.

Branching rule [Vance *et al.* '94] (cont.)

in right branch: put items r and s in different bins

- in Master Problem: set to 0 all columns that are infeasible,
- in Subproblem: add extra constraint to avoid having both items r and s

$$\begin{aligned} \max\ z_s = \quad & \sum_{i \in S} \pi_i y_i \\ \textit{subj. to} \quad & \sum_{i \in S} w_i y_i \leq W \\ & y_r + y_s \leq 1 \\ & y_i \in \{0,1\},\ \forall i \in S \end{aligned}$$

- after b branchings, b pairs of constraints are added.

Structure of subproblem is preserved in GAP, but not in BCSP.

Binary multicommodity flow problem

Flow of K commodities, indexed by k, in a graph $G = (V, A)$
u_{ij} : capacity of arc (i, j)

Commodity k has a flow of q^k, from one unique supply node and to one unique demand node. Node i has:

a positive supply of b_i^k units of commodity k, if i is one of the origin nodes for k,
a positive demand of $-b_i^k$ units of commodity k, if i is one of the destination nodes for k,
a null value, otherwise.

The flow of each commodity must be routed in a single path.

Binary multicommodity flow problem: arc flow model

Decision variables $x_{ij}^k = \begin{cases} 1, & \text{if entire flow of commodity } k \text{ uses arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$

c_{ij}^k : unit cost of arc (i,j)

$q^k c_{ij}^k$: cost of entire flow of commodity k in arc (i,j)

$$\begin{aligned} \min \quad & \sum_{k \in K} \sum_{(i,j) \in A} q^k c_{ij}^k x_{ij}^k \\ \text{subj. to} \quad & + \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k, \ , \ \forall i \in V, \forall k \in K \\ & \sum_{k \in K} q^k x_{ij}^k \leq u_{ij} \ , \ \forall (i,j) \in A \\ & x_{ij}^k \in \{0,1\}, \ \forall k, \forall (i,j) \in A \end{aligned}$$

Binary multicommodity flow problem: path model

P^k : set of paths between source node and destination node of commodity k.

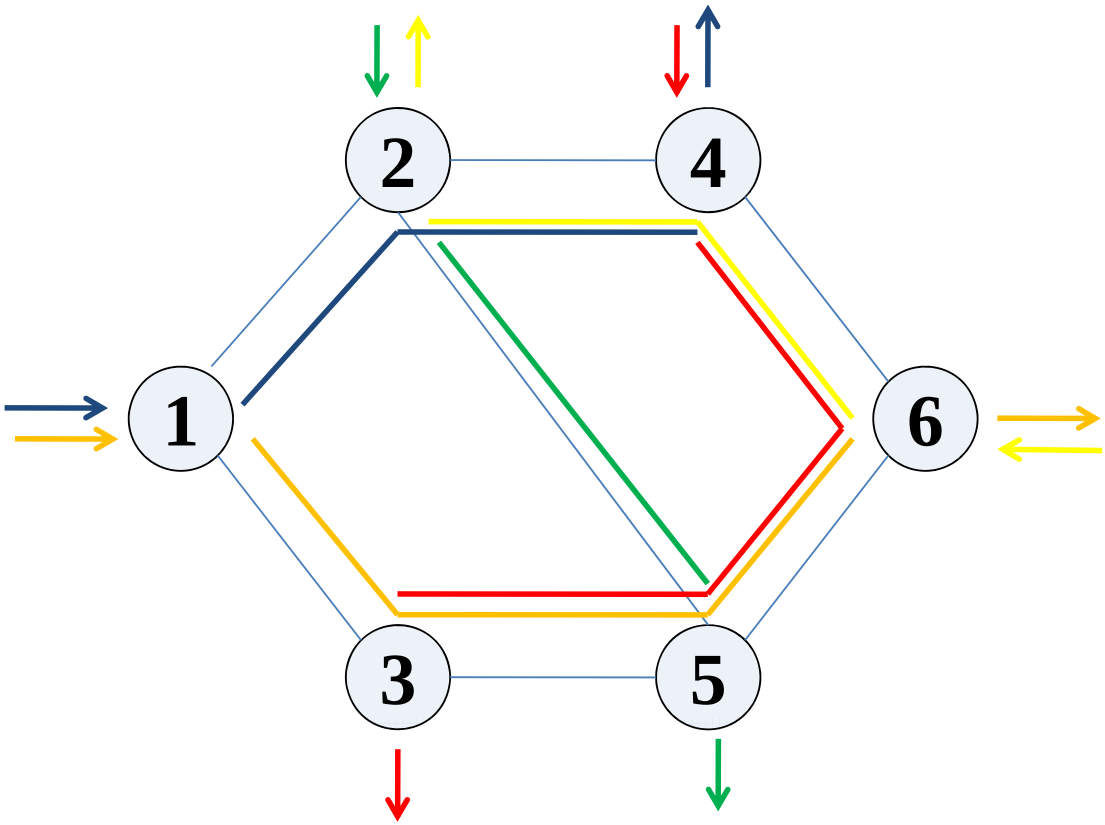
Reform. variables $y_p^k = \begin{cases} 1, & \text{if path } p \in P^k \text{ is used for commodity } k \\ 0, & \text{otherwise} \end{cases}$

c_p^k : corresponding unit cost for the path, *i.e.*, $c_p^k = \sum_{(i,j)\in A} \delta_{ij}^p c_{ij}^k$.

where $\delta_{ij}^p = \begin{cases} 1, & \text{if arc } (i,j) \text{ belongs to path } p \\ 0, & \text{otherwise} \end{cases}$

$$\begin{aligned} \min \quad & \sum_{k\in K} \sum_{p\in P^k} q^k c_p^k y_p^k \\ \text{subj. to} \quad & \sum_{k\in K} \sum_{p\in P^k} \delta_{ij}^p q^k y_p^k \le u_{ij} \ , \ \forall (i,j) \in A \\ & \sum_{p\in P^k} y_p^k = 1, \ \forall k \in K \\ & y_p^k \in \{0,1\}, \ \forall p \in P^k, \forall k \in K \end{aligned}$$

Example



Branching on variables of reformulated model

Binary multicommodity flow problem: branching on paths

- in left branch, $y_p^k = 1$ is easy to deal with:
  - commodity k is done,
  - just reduce the capacities of arcs in path p by q^k.
- in right branch, $y_p^k = 0$ forbids commodity k to use path p:
  - (shortest path) subproblem must know that it should not generate path p for commodity k,
  - which may be (and often is) the most attractive path to the subproblem after the branching constraint is added to the master problem ...

Avoiding the regeneration of variables ... to avoid a deadlock

Modify subproblem, so as to:

reject the most attractive column (if you do not want it in the master problem), the 2^{nd}, the 3^{rd}, ..., the k^{th} best columns, until an column acceptable is found.

Application: binary multicommodity flow problem:

use k^{th} best shortest path problem in subproblem.

Application: cutting stock problem:

Degraeve, Schrage' 99 : (modified) knapsack subproblem receives a list of forbidden solutions, and only generates acceptable solutions.

Binary multicommodity problem: branching on x_{ij}^k variables

x_{ij}^k are variables of the original model

- in right branch, $x_{ij}^k = 0$ forbids commodity k to use arc (i,j) : just remove arc (i,j) from (shortest path) subproblem graph.
- in left branch, $x_{ij}^k = 1$ forces commodity k to use arc (i,j) :
  - easy if there is a single constraint,
  - complicated if shortest path must go through a set of arcs, when several constraints are enforced in the node, deep in the tree.

Branching rule [Barnhart *et al.* '00]

Binary multicommodity flow problem: exclude set of arcs of commodity in one branch and the complementary set in the other branch

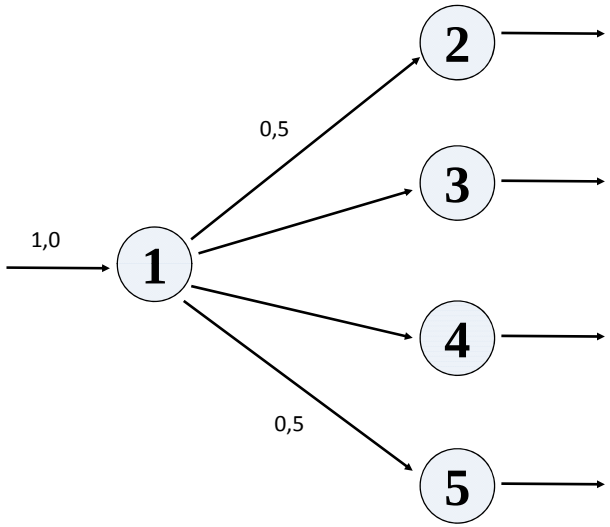
For a commodity k with fractional flows x_{ij}^k out of a node i:

- Choose a(n even) partition of the set J of successors of $i : \overline{J}$ and $J \backslash \overline{J}$,
- such that $\sum_{j \in \overline{J}} x_{ij}^k < 1$,
- and use branching constraints:
  - $\sum_{j \in \overline{J}} x_{ij}^k \leq 0$
  - $\sum_{j \in J \backslash \overline{J}} x_{ij}^k \leq 0$

Constraints are easy to enforce in the subproblem in both branches, because arcs are just removed from subproblem.

Example

- left branch: $x_{12}^k + x_{13}^k = 0$ – arcs $(1,2)$ and $(1,3)$ excluded
- right branch: $x_{14}^k + x_{15}^k = 0$ – arcs $(1,4)$ and $(1,5)$ excluded



left and right branches are not mutually disjoint:

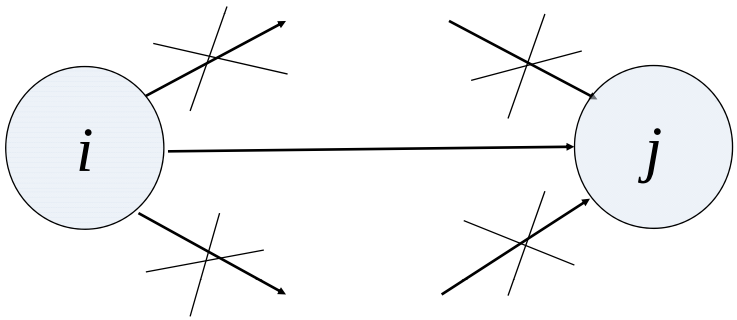
Solutions with null flow in all arcs $(1,2),(1,3),(1,4)$ and $(1,5)$ belong to both branches.

Vehicle routing with TW [Desrochers *et al.* '92]

All clients are visited once:

in left branch: cover clients i and j with the same route

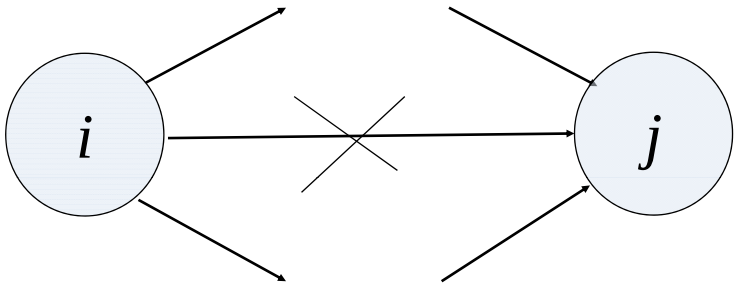
- in Subproblem network:
  - fix arc (i,j) at 1.
  - arcs $(i,k), k \neq j$ are removed
  - arcs $(l,j), l \neq i$ are removed
- in Master Problem: penalize all columns of master problem that use arcs removed in subproblem (penalty should be sufficient to drive then to 0)



Vehicle routing with TW [Desrochers *et al.* '92] (cont.)

in right branch: cover clients i and j with different routes

- in Subproblem network:
  - arc (i,j) is removed
- in Master Problem: again penalize all columns of master problem that use arcs removed in subproblem



Carpaneto and Toth's rule (1980)

- Pick a fractional variable y_r of the master problem corresponding to a route with s arcs: $v_1, v_2, \ldots, v_s, v_1$.
- Create $s+1$ branches on arc variables of the route:

$$
\begin{array}{lcl}
\text{branch } 1 & : & x_{v_1 v_2} = 0 \\
\text{branch } 2 & : & x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 0 \\
 & \ldots & \\
\text{branch } s & : & x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_{s-1} v_s} = 1,\ \ldots,\ x_{v_s v_1} = 0 \\
\text{branch } s+1 & : & x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_{s-1} v_s} = 1,\ \ldots,\ x_{v_s v_1} = 1
\end{array}
$$

- creates a polynomial number of branches
- used in J. Desrosiers, F. Soumis, M. Desrochers, Routing with Time Windows by Column Generation, Networks, 14, pp. 545–565, 1984.

Example: route with 3 arcs

Consider route: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_1$.

$$
\begin{aligned}
&\text{branch } 1: && x_{v_1 v_2} = 0 \\
&\text{branch } 2: && x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 0 \\
&\text{branch } 3: && x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 0 \\
&\text{branch } 4: && x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 1
\end{aligned}
$$

	$x_{v_1 v_2}$	$x_{v_2 v_3}$	$x_{v_3 v_1}$
	0	0	0
branch 1	0	0	1
	0	1	0
	0	1	1
branch 2	1	0	0
	1	0	1
branch 3	1	1	0
branch 4	1	1	1

Dealing with branching constraints

branch 1 : $x_{v_1 v_2} = 0$:

- remove arc $x_{v_1 v_2}$ from subproblem

branch 2 : $x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 0$:

- group trips 1 and 2 both in the master problem and in subproblem
- remove arc $x_{v_2 v_3}$ from subproblem

branch 3 : $x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 0$:

- group trips 1, 2 and 3 both in the master problem and in subproblem
- remove arc $x_{v_3 v_1}$ from subproblem

branch 4 : $x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 1$:

- route is fixed: remove trips 1, 2 and 3

Considering branching constraints explicitly

In all the previous examples, the branching constraints were not added to the Restricted Master Problem.

General strategy [Vanderbeck, Wolsey'96]:

- Consider the structure of the RMP at a given node of the Branch-and-Price tree,
- Use dual information from the branching constraints in the subproblem.

Considering branching constraints explicitly (cont.)

Restricted master problem at a given node of the branch-and-price tree:

$$
\begin{aligned}
\min z \quad = \quad & \sum_{j \in J} c_j x_j \\
\text{s. to} \quad & \sum_{j \in J} a_j x_j = b \\
& \sum_{j \in J} x_j \leq U \\
& \sum_{j \in J} x_j \geq L \\
& x_j \geq 0, \text{ and integer, } \forall j \in J,
\end{aligned}
$$

$\pi \in \mathbb{R}^m, \mu \in \mathbb{R}_-, \nu \in \mathbb{R}_+$ are the dual variables corresponding to each set of constraints, respectively.

Considering branching constraints explicitly (cont.)

Subproblem:

- there are new constraints in the Restricted Master Problem,
- it may be necessary to consider additional binary variables in the subproblem to enforce in the subproblem the dual information of the Restricted Master Problem,

Compatibility:

- if that happens, subproblem loses its structure,
- it may become a general Integer Programming Problem.

Concluding remarks

- compatibility is a crucial issue in branch-and-price
- in models with binary variables, it is often possible to implement branch-and-price without adding explicitly the constraints.

Branch-and-price algorithms (cont.)

Outline

- Branch-and-price: general integer variables
- Arc-flow model
- Application example: cutting stock problem
- Extension to the Multiple lengths cutting stock problem

Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.

Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.
- Oriented acyclic graph $G = (V, A)$.

Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.
- Oriented acyclic graph $G = (V, A)$.
- $V = \{0, 1, 2, \ldots, W\}$.

Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.
- Oriented acyclic graph $G = (V, A)$.
- $V = \{0, 1, 2, \ldots, W\}$.
- $A = \{(i,j) : 0 \leq i < j \leq W \text{ and } j - i = w_d,\ d = 1, \ldots, m\}$: length of oriented arc defines size of item.

Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.
- Oriented acyclic graph $G = (V, A)$.
- $V = \{0, 1, 2, \ldots, W\}$.
- $A = \{(i,j) : 0 \leq i < j \leq W \text{ and } j - i = w_d,\ d = 1, \ldots, m\}$: length of oriented arc defines size of item.
- Additional arcs $(k, k+1), k = 0, 1, \ldots, W-1$, correspond to loss.

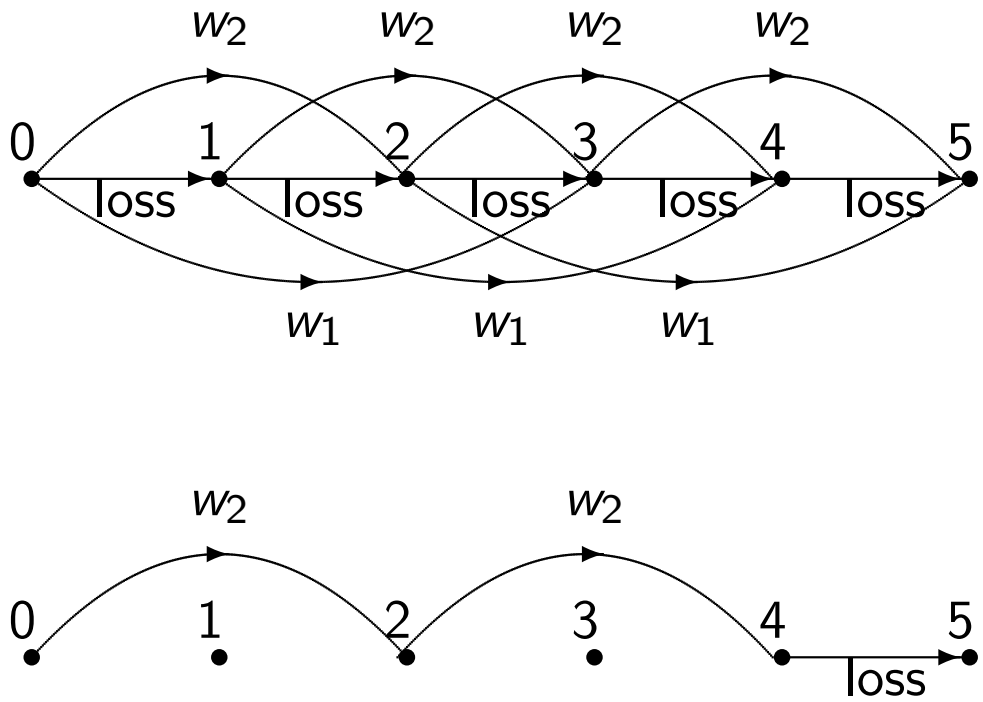
Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.
- Oriented acyclic graph $G = (V, A)$.
- $V = \{0, 1, 2, \ldots, W\}$.
- $A = \{(i,j) : 0 \leq i < j \leq W \text{ and } j - i = w_d,\ d = 1, \ldots, m\}$: length of oriented arc defines size of item.
- Additional arcs $(k, k+1), k = 0, 1, \ldots, W-1$, correspond to loss.
- Valid cutting pattern is a path between vertices 0 and W.

Cutting Stock Problem: arc flow model [VC, 1999]

- Rolls of integer capacity W and items of integer size $w_1, \ldots, w_d, \ldots, w_m$.
- Oriented acyclic graph $G = (V, A)$.
- $V = \{0, 1, 2, \ldots, W\}$.
- $A = \{(i,j) : 0 \leq i < j \leq W \text{ and } j - i = w_d,\ d = 1, \ldots, m\}$: length of oriented arc defines size of item.
- Additional arcs $(k, k+1), k = 0, 1, \ldots, W-1$, correspond to loss.
- Valid cutting pattern is a path between vertices 0 and W.
- The number of variables is $O(mW)$.

Example: rolls of width $W = 5$, items of sizes 3 and 2



Path corresponds to 2 items of size 2 and 1 unit of loss.

Arc flow model: main ideas

- Flow of one unit from vertex 0 to vertex W corresponds to one cutting pattern.
- Larger flow corresponds to the same cutting pattern in several rolls.
- Flow Decomposition property (graph G is acyclic): any flow can be decomposed in oriented paths connecting the only supply node (node 0) to the only terminal node (node W).
- Solution with integer flows is decomposed into an integer solution for the Cutting Stock Problem.

Ahuja, Magnanti, Orlin'93:

Every nonnegative arc flow can be represented as a path and cycle flow (though not necessarily uniquely) with the following two properties:
i) every oriented path with positive flow connects a deficit node to an excess node.
ii) At most $n+m$ paths and cycles have nonzero flow; out of these, at most m cycles have nonzero flow.

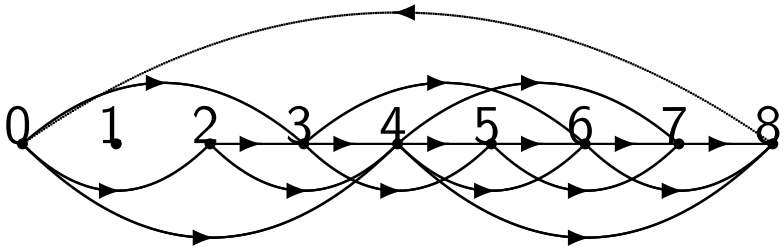
Arc flow model

Decision variables x_{ij} : flow in arc $(i,j) \equiv$ number of items of size $j-i$ placed in any roll at a distance i of the border of the roll.

$$\begin{aligned} \min \quad & z \\ \text{subj. to} \quad & + \sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = \begin{cases} -z & \text{, if } j = 0 \\ 0 & \text{, if } j = 1,\ldots,W-1 \\ z & \text{, if } j = W \end{cases} \\ & \sum_{(k,k+w_d)\in A} x_{k,k+w_d} \geq b_d \ , \ d = 1,2,\ldots,m \\ & x_{ij} \geq 0 \ \text{ and integer } \ , \ \forall (i,j) \in A \end{aligned}$$

Constraint set 1: flow conservation $\equiv$ valid cutting patterns.
Constraint set 2: sum of flows in arcs of each size $\geq$ demand.
Objective: minimize $z \equiv$ flow between vertex 0 and vertex W.

Arc flow model: example



		x_{04}	x_{48}	x_{03}	x_{36}	x_{47}	x_{02}	x_{24}	x_{35}	x_{46}	x_{57}	x_{68}	z	
node	0	-1		-1			-1						1	$= 0$
	1													$= 0$
	2						1	-1						$= 0$
	3			1	-1				-1					$= 0$
	4	1	-1			-1		1		-1				$= 0$
	5								1		-1			$= 0$
	6				1					1		-1		$= 0$
	7					1					1			$= 0$
	8		1									1	-1	$= 0$
$w_d =$	4	1	1											≥ 5
	3			1	1	1								≥ 4
	2						1	1	1	1	1	1		≥ 8

The loss arcs in the Figure are omitted in the LP model.

Equivalence with Gilmore-Gomory model

Proposition

Arc flow model is equivalent to classical Gilmore-Gomory model.

Proof: applying a DW decomposition to arc flow model gives Gilmore-Gomory model.

- Keep demand constraints in the master problem and flow constraints in the subproblem.
- Each path (cutting pattern) corresponds to an integer solution of the knapsack subproblem.
- Each path is part of a circulation flow (includes the z variable), which is an extreme ray of the subproblem.
- Null solution is the only extreme point.
- Otherwise, there are extreme rays: no convexity constraint.

Symmetry

A model has symmetry when different solutions, in terms of the values of the decision variables, correspond to the same physical solution in the real system.

Example Kantorovich model: decision variable $x_{ij} = 1$, if item j is placed in roll i.

Two solutions: items 1 and 2 are placed in a bin, and the items 3 and 4 in the other:

solution 1: $x_{11} = x_{12} = x_{23} = x_{24} = 1$;

solution 2: $x_{21} = x_{22} = x_{13} = x_{14} = 1$.

correspond to the same packing. □

Example Arc-flow model:

solution 1: $x_{03} = x_{35} = x_{57} = x_{78} = 1$;

solution 2: $x_{02} = x_{24} = x_{47} = x_{78} = 1$.

correspond to the same cutting pattern. □

Reduction of symmetry

Criterion

An arc of size w_e, denoted as $x_{k,k+w_e}$, can only have its tail at a node k that is the head of another arc of size w_d, $x_{k-w_d,k}$, for $w_d \geq w_e$, or, else, at node 0, i.e., the left border of the roll.

If a cutting pattern has loss, it will appear at the end of the roll:

Criterion

Arcs of loss $x_{k,k+1}$ may be set to zero, for $k < w_m$.

In a cutting pattern, the number of consecutive arcs of a given size should be less than or equal to the number of demanded items of that size.

Criterion

Given a node k that is the head of another arc of size w_d $(w_d > w_e)$ or $k = 0$, the only valid arcs of size w_e are those that start at nodes $k + sw_e, s = 0, 1, 2, \ldots, b_e - 1$ and $k + sw_e \leq W$, being b_e the demand of items of size w_e.

Symmetry does not arise in the following solution methodology.

Branch-and-price methodology for CSP

Master Problem: Gilmore-Gomory model + branching constraints based on arc flow variables.

Finding a fractional arc flow variable for branching:

Find arc flows x_{pq} reading Gilmore-Gomory variables:

- Assumption: items in cutting pattern placed by decreasing size.
- Cutting pattern contributes x_j to flow of original variable x_{pq} if there is an item of size $q-p$ beginning at p,
- *i.e.*, value of the flow x_{pq} is given by:

$$x_{pq} = \sum_{j \in \bar{J}} x_j$$

Example: first branching constraint

- Fractional optimal solution of the linear relaxation:

A_j	A_1	A_3	A_4
	4	2	3
		2	
	4	2	3
		2	2
$x_j =$	2.5	1.5	2.0

- Flows in arcs: $x_{04} = 2.5$, $x_{48} = 2.5$, $x_{03} = 2.0$, $x_{36} = 2.0$, $x_{02} = 1.5$, $x_{24} = 1.5$, $x_{46} = 1.5$, and $x_{68} = 3.5$.
- First branching constraint: $x_{04} \geq 3$.

Branching scheme

- Branching rule (simple): create 2 branches:

$$x_{ij} \leq \lfloor x_{ij} \rfloor$$

and

$$x_{ij} \geq \lceil x_{ij} \rceil$$

- Variable selection: fractional largest item size, closer to the top border of the roll.
- Search: depth-first search ($\geq$ branch explored first).
- Branching constraint respects to a single arc in position (i,j).
- Branching constraint only affects the cutting patterns with an arc in position (i,j).

Restricted master problem in node w of the search tree

$$
\begin{array}{lll}
min & \sum_{j\in J} x_j & \\
s.\ to & \sum_{j\in J} a_{dj}x_j \geq b_d,\ d=1,2,\ldots,m & \leftarrow \textbf{GG model} \\
 & \sum_{j\in J} \delta_j^l x_j \leq \lfloor x_{ij}^l \rfloor\ ,\ \forall l \in G^w & \leftarrow \textbf{branching constraints} \\
 & \sum_{j\in J} \delta_j^l x_j \geq \lceil x_{ij}^l \rceil\ ,\ \forall l \in H^w & \\
 & x_j \geq 0,\ \forall j \in J, &
\end{array}
$$

G^w, H^w : sets of branching constraints of the types $\leq$ and $\geq$, respectively.
x_{ij}^l : the fractional values of flow $0 < x_{ij}^l < b_d$.
$\delta_j^l = 1$, if the arc $(i, i+w_d) \in$ cutting pattern j; or 0, otherwise.

Note: CSP has general integer variables

Most applications have binary variables.

Dual information for the subproblem

- Prize / penalty from a branching constraints of type $\geq$ and $\leq$, respectively, only change reduced cost of one arc in the subproblem.
- In node w, the reduced cost of arc (i,j) is

$$\overline{c}_{ij} = \pi_d - \sum_{l \in G^w_{(i,j)}} \mu_l + \sum_{l \in H^w_{(i,j)}} \nu_l,$$

  $G^w_{(i,j)} \subseteq G^w$, $H^w_{(i,j)} \subseteq H^w$: sets of branching constraints on arc (i,j).
- Structure of subproblem remains unchanged during branch-&-price.
- Subproblem is solved using dynamic programming (pseudopolynomial).

Restricted problem: first node of branch-and-price tree

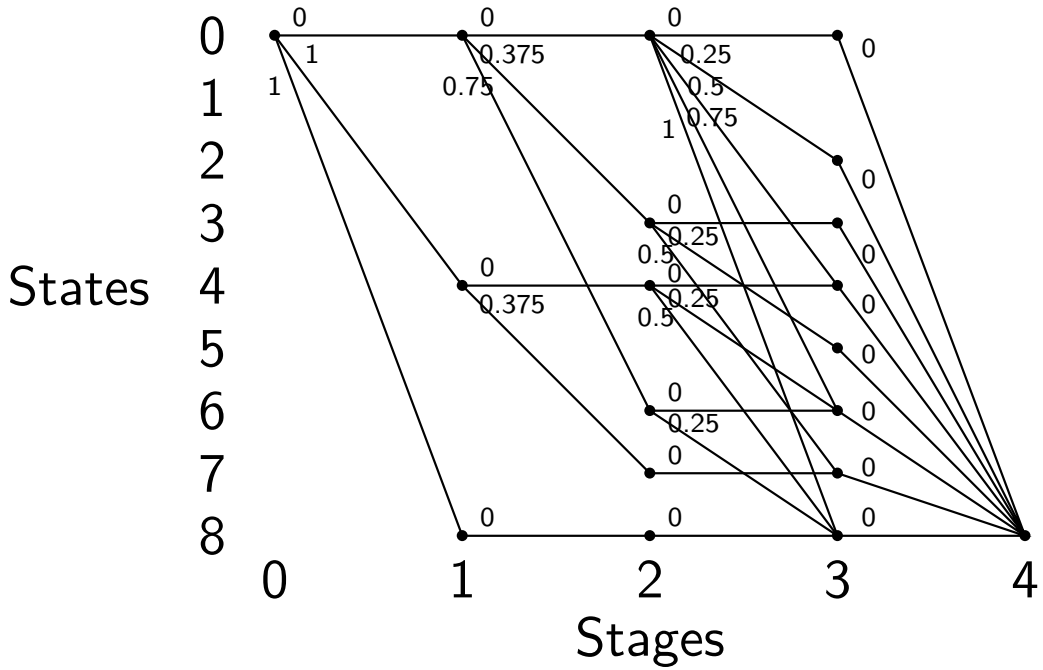
Insert branching constraint $x_{04} \geq 3$, and reoptimize:

	x_1	x_2	x_3	x_4			dual
$w_d =$ 4	2				$\geq$	5	0.0
3		2		2	$\geq$	4	0.375
2			4	1	$\geq$	8	0.25
$x_{04} \geq 3$	1				$\geq$	3	1.0
min	1	1	1	1			

primal	3.0	0.0	1.5	2.0	$z^1 =$	6.5

Dual info: prize of 1 associated to branching constraint $x_{04} \geq 3$..

Subproblem: first node of branch-and-price tree



In stage 0, placing 1 or 2 items has a contribution equal to 1.
First decision: arc (0,4); second decision: arcs (0,4) and (4,8).
Optimal solution: 1 item of size 4 and 2 items of size 2 (value=1.5).

Optimal integer solution

The new column has a 1 in the branching constraint (sum of flows in arc (0,4) across all cutting patterns must be ≥ 3).
After reoptimizing:

	x_1	x_2	x_3	x_4	x_5			dual
$w_d = 4$	2				1	$\geq$	5	0.5
3		2		2		$\geq$	4	0.375
2			4	1	2	$\geq$	8	0.25
$x_{04} \geq 3$	1				1	$\geq$	3	0.0
min	1	1	1	1	1			

primal	2.0	0.0	1.0	2.0	1.0	$z^* =$	6.0

The solution is integer, with a value equal to the LP relaxation.
Optimal solution!

Multiple lengths cutting stock problem (MLCSP)

Multiple stock lengths available, instead of a single roll size.
Proposed by Gilmore-Gomory'63 (machine balance problem).
Many heuristic approaches: Chu,La'2001, Holthaus'2002
Few exact solution approaches: Monaci'2002, Belov,Scheithauer'2002

Equivalent counterpart in bin-packing literature is the Variable sized bin-packing problem (VSBPP).

Multiple lengths cutting stock problem: model

a_{ikr} : number of items of width w_i obtained in stock length k using pattern r.
λ_{kp} : number of times a pattern p from stock length k is cut.
P_k : set of feasible cutting patterns of stock length k, $k = 1, \ldots, K$.

$$\begin{aligned}
\min \quad & \sum_{k=1}^{K} \sum_{p \in P^k} W_k \lambda_{kp} \\
\text{subj. to} \quad & \sum_{k=1}^{K} \sum_{p \in P^k} a_{ikp} \lambda_{kp} \geq b_i, \ i = 1, \ldots, m, \\
& \sum_{p \in P^k} \lambda_{kp} \leq B_k, \ k = 1, \ldots, K, \\
& \lambda_{kp} \geq 0 \text{ and integer}, \ k = 1, \ldots, K, \ p \in P^k.
\end{aligned}$$

constraint set 1: demand constraints.
constraint set 2: availability constraints on each stock length.
(Gilmore-Gomory'63)

Multiple lengths cutting stock problem: example

Stock : availabilities $\mathscr{B} = (B_1, B_2, B_3)$, sizes $\mathscr{W} = (9, 6, 5)$.
Demand: quantities $b = (20, 10, 20)$, sizes $w = (4, 3, 2)$.
Sum of item sizes: $\sum_i w_i b_i = 80 + 30 + 40 = 150$.
Available capacity of stock lengths: $\sum_k W_k B_k = 9B_1 + 6B_2 + 5B_3$.

	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	λ_{16}	λ_{17}	λ_{21}	λ_{22}	λ_{23}	λ_{24}	λ_{31}	λ_{32}	λ_{33}	
$w_i =$ 4	2	1	1					1				1			≥ 20
3		1		3	2	1			2	1			1		≥ 10
2		1	2		1	3	4	1		1	3		1	2	≥ 20
$W_k =$ 9	1	1	1	1	1	1	1								$\leq B_1$
6								1	1	1	1				$\leq B_2$
5												1	1	1	$\leq B_3$
min	9	9	9	9	9	9	9	6	6	6	6	5	5	5	

Arc-flow model (multiple lengths cutting stock problem)

$$\min \sum_{k=1}^{K} W_k z_k$$

$$subj.\ to$$

$$-\sum_{(d,e)\in A'} x_{de} + \sum_{(e,f)\in A'} x_{ef} = \begin{cases} \sum_{k=1}^{K} z_k & , \text{ if } e = 0 \\ -z_k & , \text{ for } e = W_k, k = 1,\ldots,K \\ 0 & , \text{ otherwise} \end{cases}$$

$$\sum_{(d,d+w_i)\in A'} x_{d,d+w_i} \geq b_i \ , \quad \forall i \in I$$

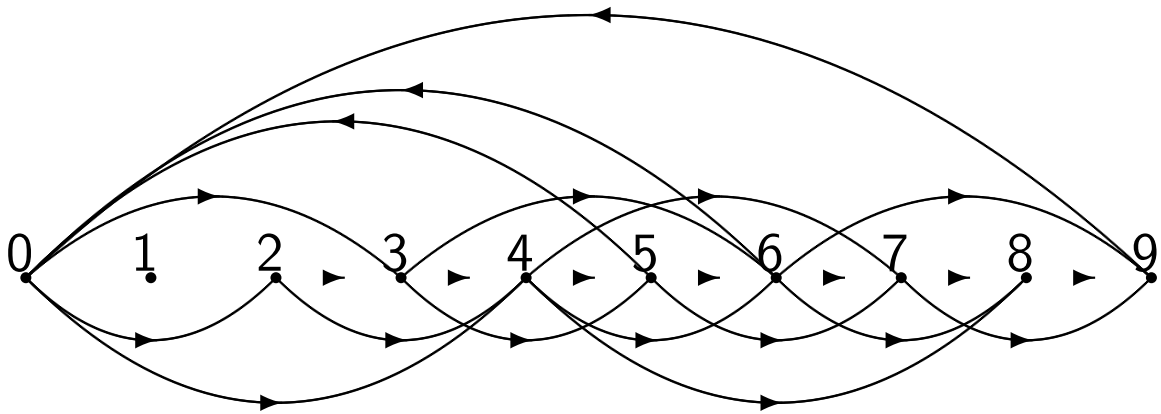
$$z_k \leq B_k \ , \ k = 1,\ldots,K$$

$$x_{de} \geq 0 \text{ and integer} \ , \ \forall (d,e) \in A'$$

$$z_k \geq 0 \text{ and integer} \ , \ k = 1,\ldots,K$$

Example (multiple lengths cutting stock problem)

Stock : availabilities $\mathscr{B} = (B_1, B_2, B_3)$, sizes $\mathscr{W} = (9, 6, 5)$.
Demand: quantities $b = (20, 10, 20)$, sizes $w = (4, 3, 2)$.
Sum of item sizes: $\sum_i w_i b_i = 80 + 30 + 40 = 150$.
Available capacity of stock lengths: $\sum_k W_k B_k = 9B_1 + 6B_2 + 5B_3$.



LP model (multiple lengths cutting stock problem)

	z_1	z_2	z_3	x_{04}	x_{48}	x_{03}	x_{36}	x_{47}	x_{69}	x_{02}	x_{24}	x_{35}	x_{46}	x_{57}	x_{68}	x_{79}	x_{23}	x_{34}	x_{45}	x_{56}	x_{67}	x_{78}	x_{89}		
node 0	1	1	1	-1		-1				-1														=	0
1																								=	0
2										1	-1						-1							=	0
3						1	-1					-1					1	-1						=	0
4				1	-1			-1			1		-1					1	-1					=	0
5		-1										1		-1					1	-1				=	0
6	-1						1		-1				1		-1					1	-1			=	0
7								1						1		-1					1	-1		=	0
8					1										1							1	-1	=	0
9	-1								1							1							1	=	0
$w_i =$ 4				1	1																			≥	20
3						1	1	1	1															≥	10
2										1	1	1	1	1	1	1								≥	20
$W_k =$ 9	1																							≤	B_1
6		1																						≤	B_2
5			1																					≤	B_3
min	9	6	5																						

Lower bounds z_{IP}^w for MLCSP at node w of branching tree

z_{LP}^w : LP bound at node w of branch-and-price tree.
Any MLCSP solution uses a combination of stock rolls with integer lengths.
Stronger integer lower bound z_{IP}^w for node w : smallest integer combination ($\geq z_{LP}^w$) of modified list of stock lengths available.
Modified list of stock lengths available (branching constraints on z arcs):
i) $z_k \geq l_k^w$: remove l_k^w rolls of width W_k from list, and consider them separately.
ii) $z_k \leq u_k^w$: use list of u_k^w rolls instead of B_k, otherwise $u_k^w = B_k$.

$$
\begin{aligned}
z_{IP}^w = \min \quad & \sum_{k=1}^{K} W_k y_k + \sum_{k=1}^{K} W_k l_k^w \\
\text{subj. to} \quad & \sum_{k=1}^{K} W_k y_k \geq \lceil z_{LP}^w \rceil - \sum_{k=1}^{K} W_k l_k^w, \\
& y_k \leq u_k^w - l_k^w, \ k = 1, \ldots, K, \\
& y_k \geq 0 \text{ andinteger}, \ k = 1, \ldots, K.
\end{aligned}
$$

Level Cuts for MLCSP

Use z_{IP}^w to enforce "Level Cuts" in one of the following ways:

1. trim loss has to appear somewhere in the cutting plan:

$$\sum_{k=1}^{K} \sum_{p \in P_k} \left(W_k - \sum_{i=1}^{m} w_i a_{ikp} \right) \lambda_{kp} \geq z_{IP}^w - \sum_{i=1}^{m} w_i b_i.$$

2. use, at least, z_{IP}^w length of rolls:

$$\sum_{k=1}^{K} \sum_{p \in P_k} W_k \lambda_{kp} \geq z_{IP}^w.$$

Dual information easily transferable to subproblem.

Feasibility cuts for MLCSP [Vanderbeck'99]

m cuts are derived, one for each item size.
Only feasibility cuts that depend on a single item size: dual info variables easily reported to the pricing subproblems.
If the availability of the largest roll $k = 1$ is enough to cut all the items of size w_i, for $i = 1, \ldots, m$, we will have

$$\sum_{k=1}^{K} \sum_{p \in P_k : a_{ikp} > 0} \lambda_{kp} \geq \left\lceil \frac{b_i}{\left\lfloor \frac{W_1}{w_i} \right\rfloor} \right\rceil .$$

Otherwise, if $\left\lceil \frac{b_i}{\left\lfloor \frac{W_1}{w_i} \right\rfloor} \right\rceil > B_1$, and the availability of stock rolls with length W_2 is enough to cut the remaining items,

$$\sum_{k=1}^{K} \sum_{p \in P_k : a_{ikp} > 0} \lambda_{kp} \geq B_1 + \left\lceil \frac{b_i - B_1 \left\lfloor \frac{W_1}{w_i} \right\rfloor}{\left\lfloor \frac{W_2}{w_i} \right\rfloor} \right\rceil ,$$

and so forth.

Implementation issues (MLCSP)

Branching scheme

- Branch first on larger fractional z arcs.
- Use branching information to improve lower bounds (Level cuts).
- Then, branch on x_{de} arcs.

Subproblem

- Single dynamic programming recursion solves subproblems for all lengths.

Cutting Stock Problem: some computational results

CSP: problems with 1 large roll width and m =200 item different sizes solved in reasonable time (triplet instances) [VC, 1999].

Multiple lengths CSP: problems with K different large roll widths (instances from literature) [Cláudio Alves, VC, 2008].

	K	m	av. time
	5	100	$\approx$ 1 sec.
	15	25	$\approx$ 1 sec.
Hard instances $\rightarrow$	5	100	$\approx$ 30 sec.

MLCSP - Computational results: summary

Branch-and-price-and-cut with simple rounding heuristic compares favorably to other approaches (results with dual cuts shown in Part V):

	K	m	av. time	
largest group inst. (Monaci)	5	100	$\approx$ 4 sec.	
our instances	15	25	$\approx$ 1 sec.	
hard instances (Belov) $\rightarrow$	5	100	$\approx$ 30 sec.	(for 45 instances solved; 5 unsolved in time limit: 900 sec. optimality gap $\leq$ 0.0025%)

K : different large roll widths (Alves'PhD2005, Alves,VC'2006).

300 instances (Monaci'2002 - Combinatorial enumeration)
Monaci solved 78% of the instances (time limit: 900 seconds).
50 hard instances (Belov'2002 - Column generation with Chvátal-Gomory cutting planes and elaborate heuristic)
Belov's approach solves less instances and takes more time (85 sec. vs. 30 sec.)

Concluding remarks

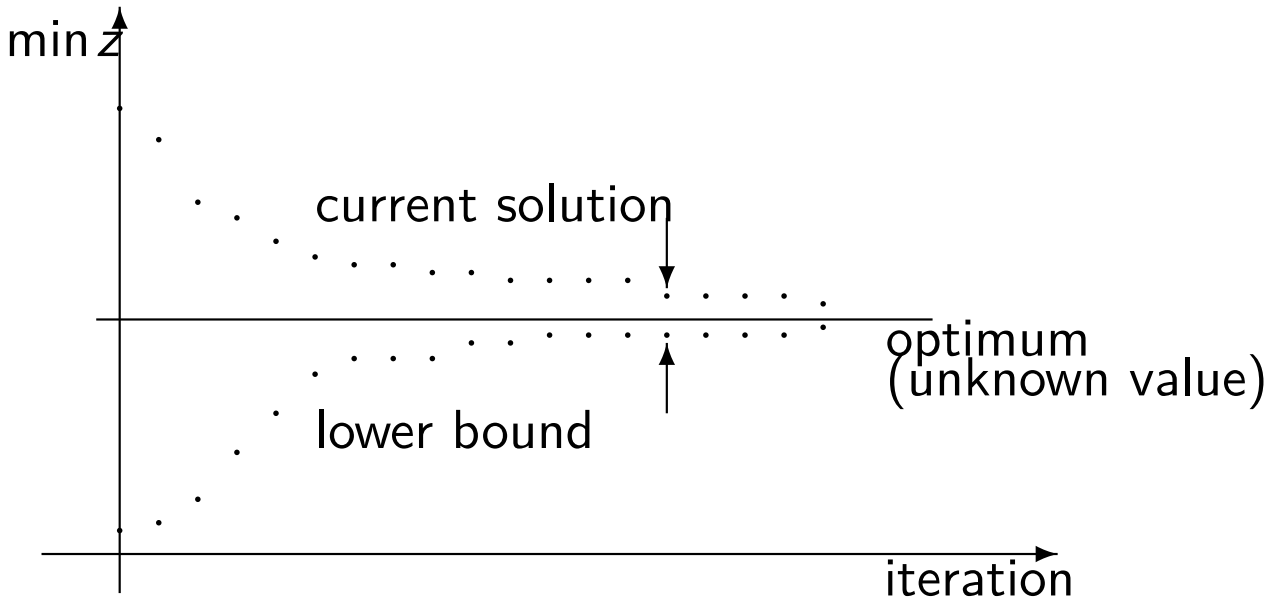
- Some heuristics provide good results for CSP; often optimal solutions.
- Heuristics do not solve optimally the MLCSP so easily.
- (GG + arc-flow) methodology is better than arc-flow solely, even with dynamic constraint generation (less symmetry and smaller size of the LP basis).

Stabilization

Outline

- Primal and dual perspectives
- Stabilizing terms: examples
- Degeneracy and perturbation
- Perfect Dual Information
- (Weak and deep) dual-optimal inequalities
- Application: cutting stock problem

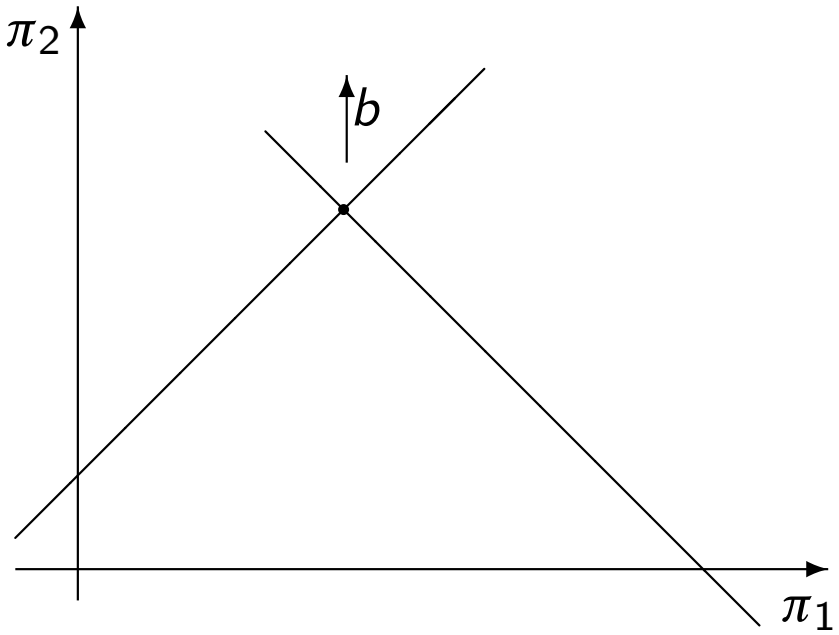
Acceleration of column generation



Slow convergence: large changes in the values of the dual variables, which oscillate from one iteration to the next.
Degeneracy: in many iterations, adding new columns to restricted master problem does not improve objective value.

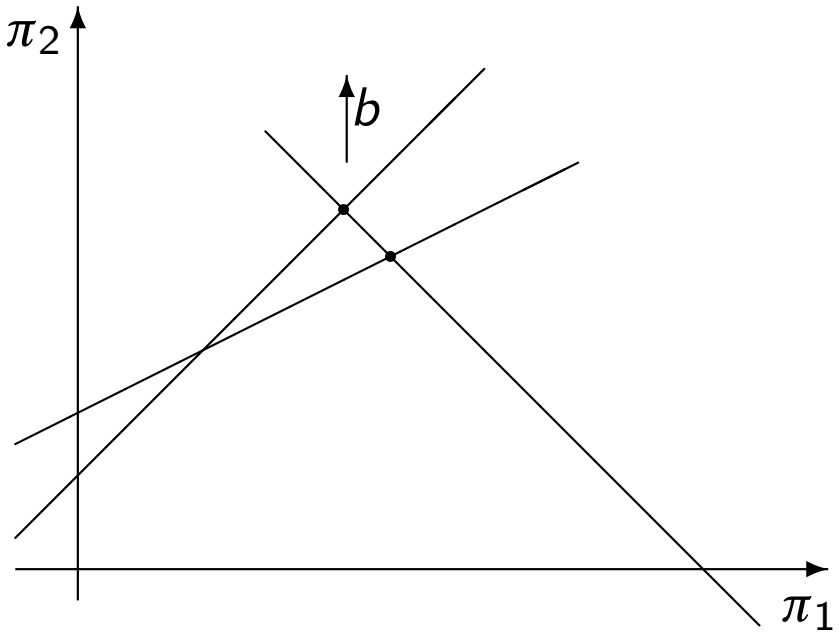
Instability of the values of the dual variables

- Dual objective: maximize dual function πb, with gradient b.
- Domain is successively restricted by adding dual constraints.
- π_2 gets smaller at each iteration (πb also does).
- π_1 oscillates until optimum dual solution is reached.



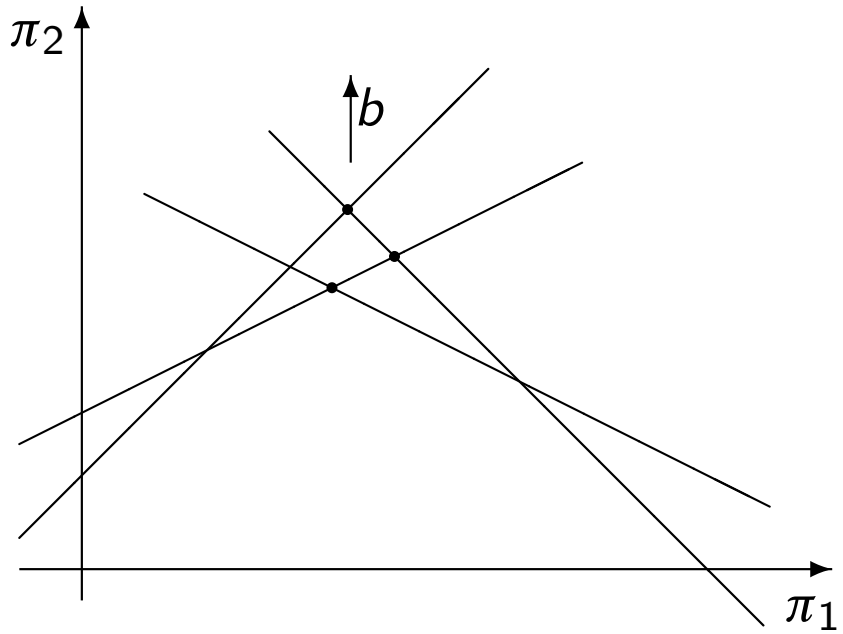
Instability of the values of the dual variables

- Dual objective: maximize dual function πb, with gradient b.
- Domain is successively restricted by adding dual constraints.
- π_2 gets smaller at each iteration (πb also does).
- π_1 oscillates until optimum dual solution is reached.



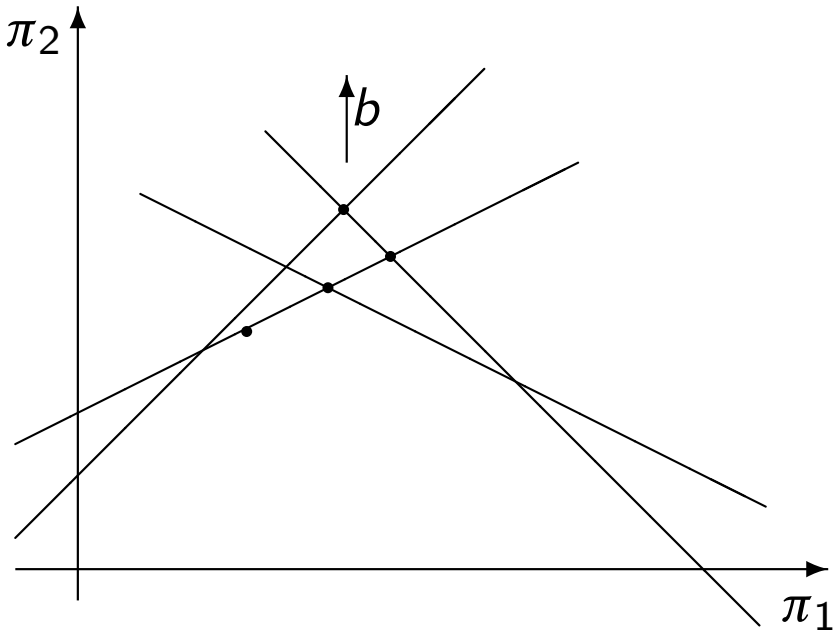
Instability of the values of the dual variables

- Dual objective: maximize dual function πb, with gradient b.
- Domain is successively restricted by adding dual constraints.
- π_2 gets smaller at each iteration (πb also does).
- π_1 oscillates until optimum dual solution is reached.



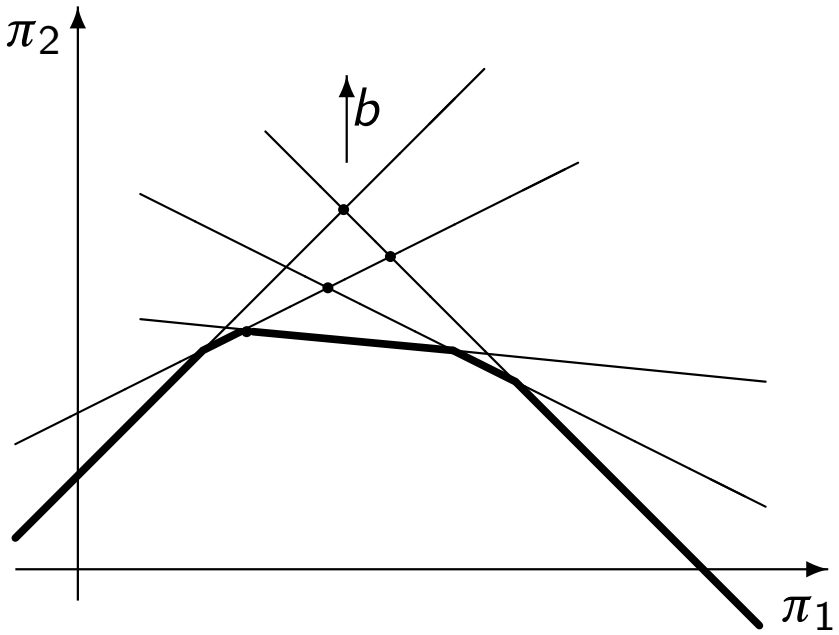
Instability of the values of the dual variables

- Dual objective: maximize dual function πb, with gradient b.
- Domain is successively restricted by adding dual constraints.
- π_2 gets smaller at each iteration (πb also does).
- π_1 oscillates until optimum dual solution is reached.



Instability of the values of the dual variables

- Dual objective: maximize dual function πb, with gradient b.
- Domain is successively restricted by adding dual constraints.
- π_2 gets smaller at each iteration (πb also does).
- π_1 oscillates until optimum dual solution is reached.



Acceleration of column generation: motivation

Restricting the dual space may accelerate column generation.
Better convergence: smaller number of attractive columns in subproblem.
Less degeneracy: alternative dual solutions ≡ degenerate primal solutions.

How to do it [VC, 2005]:
Add valid dual cuts to the model before starting column generation.

Simple example

Restricting the dual space by setting lower bounds on dual variables:
GilmoreGomory'61: for any optimal primal solution with slack for the CSP, there is an alternative optimal primal solution without slack.
What to do do: allow solutions with slack, substituting primal $=$ for $\geq$ constraints.
Faster convergence: at a given restricted master problem, there may be a solution with slack better than all solutions without slack.
Dual perspective: dual variables are restricted to be ≥ 0, instead of unrestricted.

Same happens in many practical applications, when it is valid to work with set covering instead of set partitioning formulations.

Column generation: dual perspective

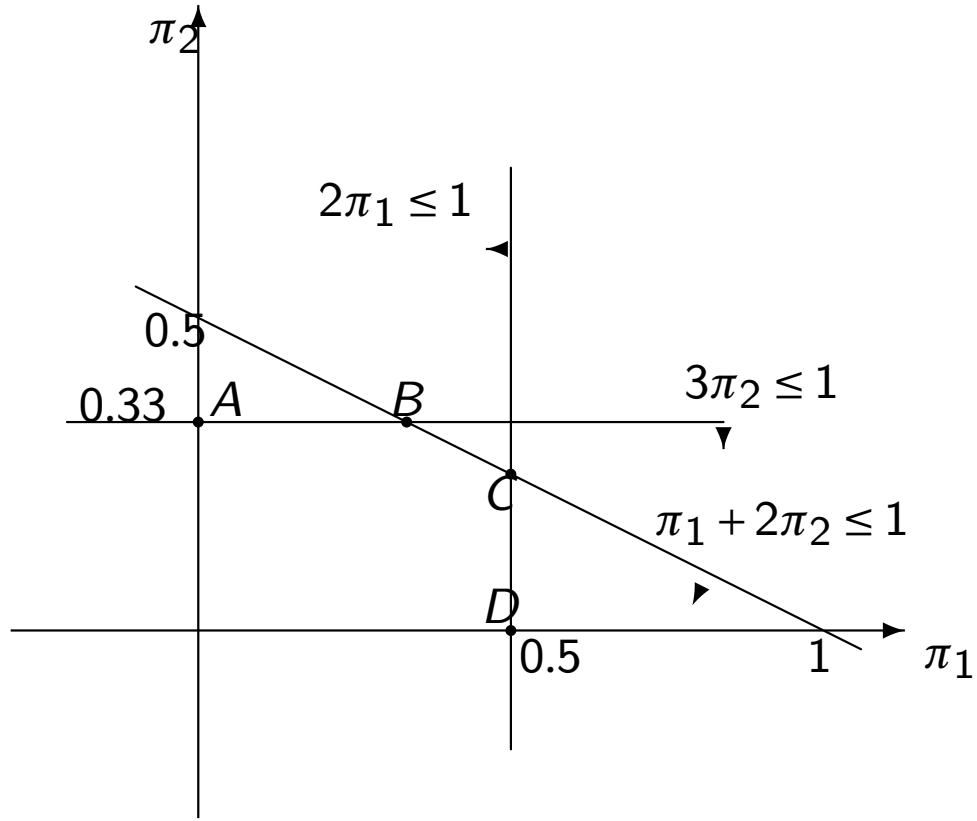
Cutting plane algorithm: adding a column in the primal is equivalent to adding a cut in the dual.

$$(Primal\ ) \quad \begin{array}{ll} \min & cx \\ s.t. & Ax \geq b \\ & x \geq 0 \end{array} \qquad (Dual) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \end{array}$$

CSP Example: rolls of width 10, items of size 4 and 3

$$(Primal\ ) \quad \begin{array}{lrrrl} \min & 1x_1 & +1x_2 & +1x_3 & \\ s.t. & 2x_1 & +1x_2 & & \geq b_1 \\ & & +2x_2 & +3x_3 & \geq b_2 \\ & x_1, & x_2, & x_3 & \geq 0 \end{array} \qquad (Dual) \quad \begin{array}{lrrl} \max & b_1\pi_1 & +b_2\pi_2 & \\ s.t. & 2\pi_1 & & \leq 1 \\ & 1\pi_1 & +2\pi_2 & \leq 1 \\ & & 3\pi_2 & \leq 1 \\ & \pi_1, & \pi_2 & \geq 0 \end{array}$$

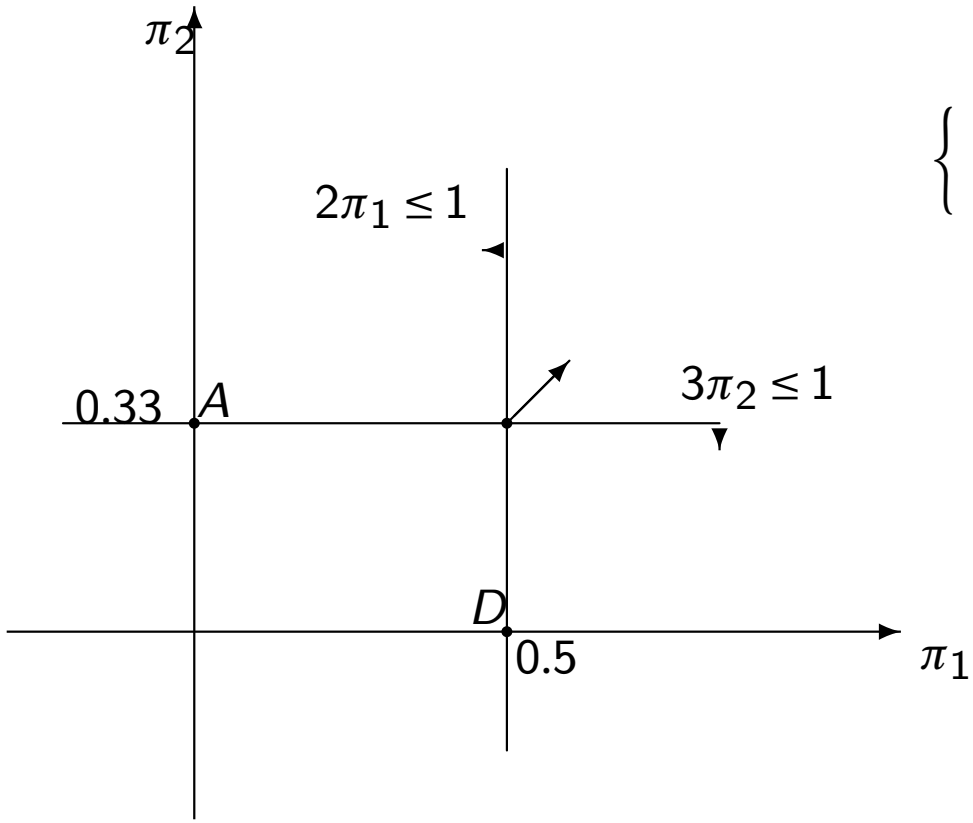
Dual space of CSP: rolls of size 10, items of size 4 and 3



knapsack:

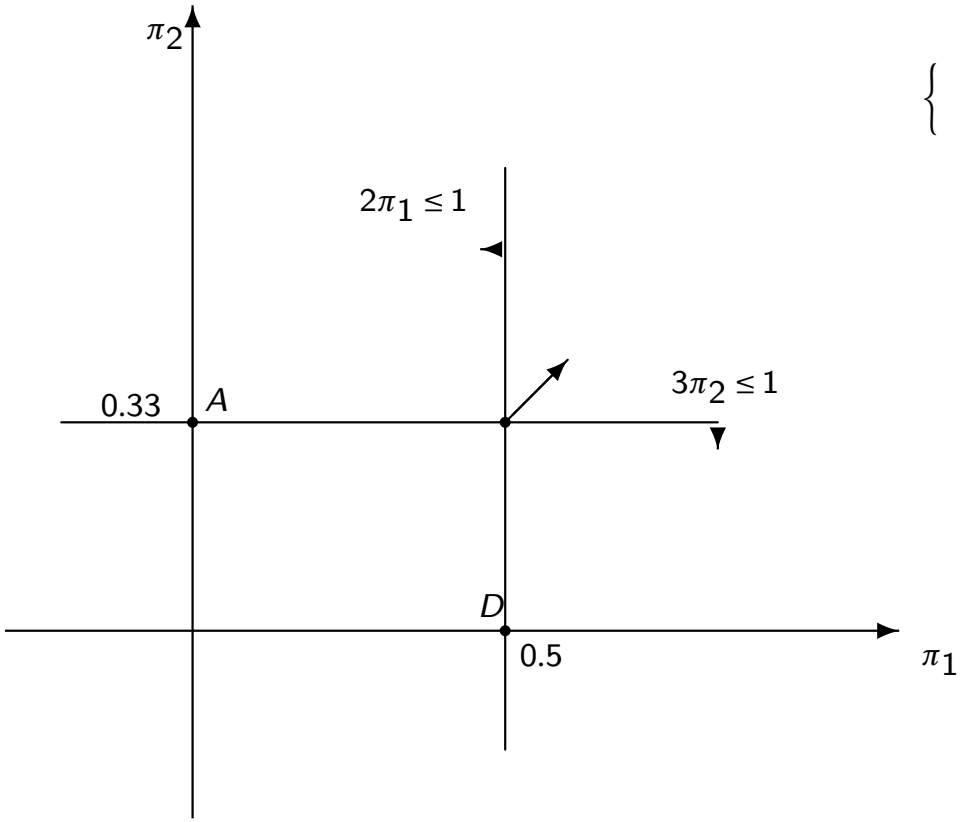
$K = \{(y_1, y_2) : 4y_1 + 3y_2 \leq 10, y_1, y_2 \geq 0 \text{ and integer}\} = \{(2,0),(1,2),(0,3)\}$

Cutting plane algorithm for dual of CSP: starting solution



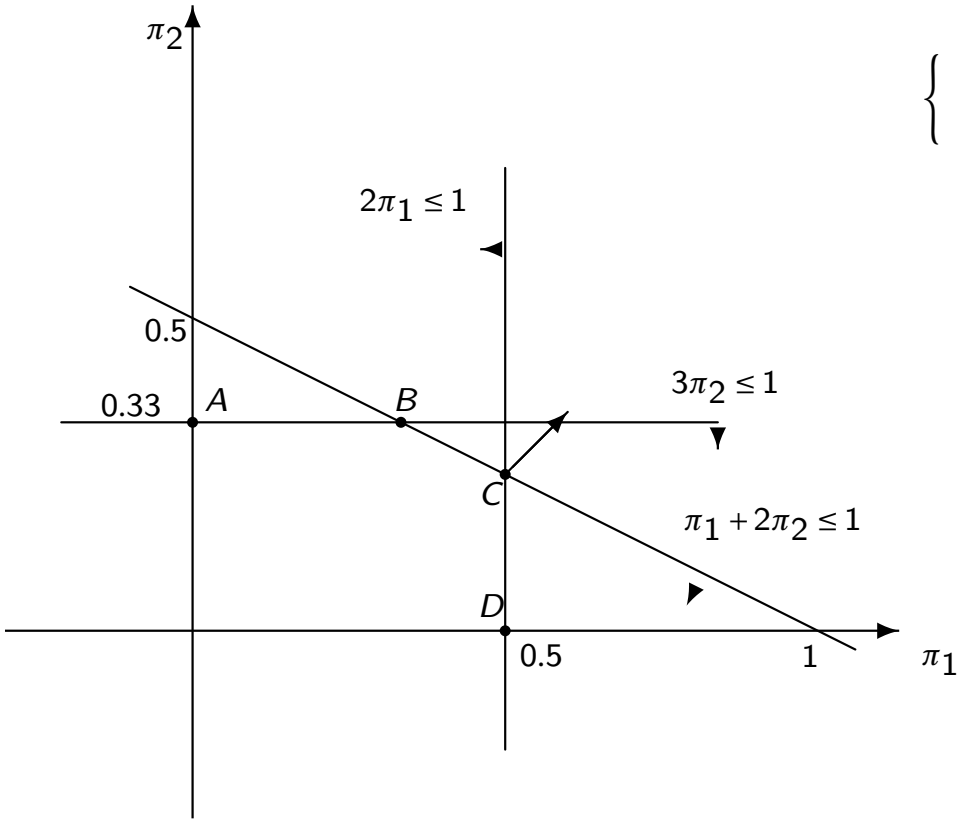
$$\begin{cases} 2\pi_1 & \leq 1 \\ 3\pi_2 & \leq 1 \end{cases}$$

Dual space of CSP: first iteration



$$\begin{cases} 2\pi_1 & \leq 1 \\ 3\pi_2 & \leq 1 \end{cases}$$

Dual space of CSP: second iteration



$$\begin{cases} 2\pi_1 & & \leq 1 \\ & 3\pi_2 & \leq 1 \\ 1\pi_1 & +2\pi_2 & \leq 1 \end{cases}$$

Dual cuts

$Ax = b$ is column generation model.

$$(P) \quad \begin{array}{ll} \min & cx \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad (D) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \end{array}$$

Adding a set of inequalities to the dual problem, $\pi D \leq d$, we get the extended primal-dual pair:

$$(P^e) \quad \begin{array}{ll} \min & cx + dy \\ s.t. & Ax + Dy = b \\ & x, y \geq 0 \end{array} \qquad (D^e) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \\ & \pi D \leq d \end{array}$$

Usually, restricting the dual $\equiv$ relaxing the primal.
In this case, that does not happen.

Assumption

Assume that we can map any solution $(\overline{x}, \overline{y})$ of the extended model to a solution $\overline{\overline{x}}$ that is valid in the original space, i.e., $\overline{\overline{x}} \in \mathscr{X} = \{x : Ax = b, x \geq 0\}$, *and has the same objective value, i.e.,* $c\overline{\overline{x}} = c\overline{x} + d\overline{y}$.

Solve the extended model, and eventually recover an optimal solution to the original problem.

Proposition

Under Assumption 1, mapping the optimal solution of the extended model $(\overline{x}^*, \overline{y}^*)$ gives a solution $\overline{\overline{x}}^*$ that is optimal to the original problem.

Proof: Let z_P^* and $z_{P^e}^*$ be the optimal values of problems P and P^e, respectively. Clearly, $z_{P^e}^* \leq z_P^*$. Let $(\overline{x}^*, \overline{y}^*)$ be the optimal solution of problem P^e. Then, $\overline{\overline{x}}^* \in \mathscr{X}$ and $c\overline{\overline{x}}^* = c\overline{x}^* + d\overline{y}^* = z_{P^e}^* \leq z_P^*$, which means that $\overline{\overline{x}}^*$ is optimal to the original problem. It follows that $z_{P^e}^* = z_P^*$. □

Insight

Dual inequalities may effectively cut portions of the space of the dual problem D ($\pi A \leq c$), but

Corollary

Under Assumption 1, the dual inequalities do not cut all optimal dual solutions of the original problem.

Proof: Let z_D^* and $z_{D^e}^*$ be the optimal values of problems D and D^e, respectively. Suppose that all optimal dual solutions were cut. Then, $z_{D^e}^* < z_D^*$, and, by the strong duality theorem, $z_{P^e}^* < z_P^*$, contradicting the previous Proposition. □

That also happens, if, at the optimum of the extended model, the dual inequality is obeyed with slack, that is, $\pi D < d$.

A family of valid dual cuts

Proposition

For any width w_i, and a set S of item widths, indexed by s, such that $\sum_{s \in S} w_s \leq w_i$, the dual cuts

$$-\pi_i + \sum_{s \in S} \pi_s \quad \leq \quad 0, \quad \forall i, S,$$

are valid inequalities to the space of optimal solutions of the dual of the cutting stock problem.

Proof.

(contradiction): there would be an attractive cutting pattern. □

Primal point of view: an item of size w_i can be cut, and used to fulfill the demand of smaller orders, provided the sum of their widths is $\leq w_i$.

Example

Combining a cutting pattern and a valid dual cut gives a new cutting pattern.

$W = 100$	A_1		D_1		A_1		A_1^{new}
25	2	+	0	-->	2	+	2
10	4		-1		4		0
6	1		1		1		5
3	0		1		0		4
2	2		0		2		2
x_j	0.3		0.8		0.1		0.2

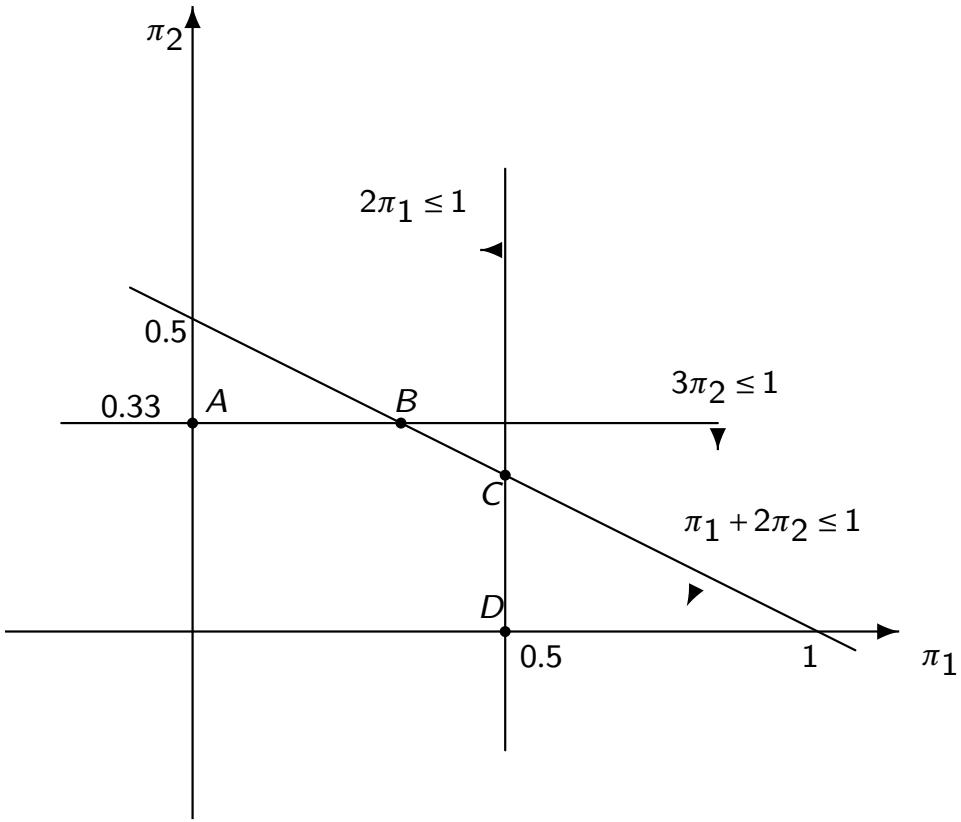
Implementation issues

- Exponential number of cuts of this family.
- Use only cuts from sets S of small cardinality.
- Sets of size 1 and 2 provide a polynomial number $O(m^2)$ of cuts.

Cuts selected:

- **Cuts of Type 1:** $-\pi_i + \pi_{i+1} \leq 0, \quad i = 1, 2, \ldots, m-1$
- **Cuts of Type 2:** $-\pi_i + \pi_j + \pi_k \leq 0, \quad \forall i, j, k : w_i \geq w_j + w_k$

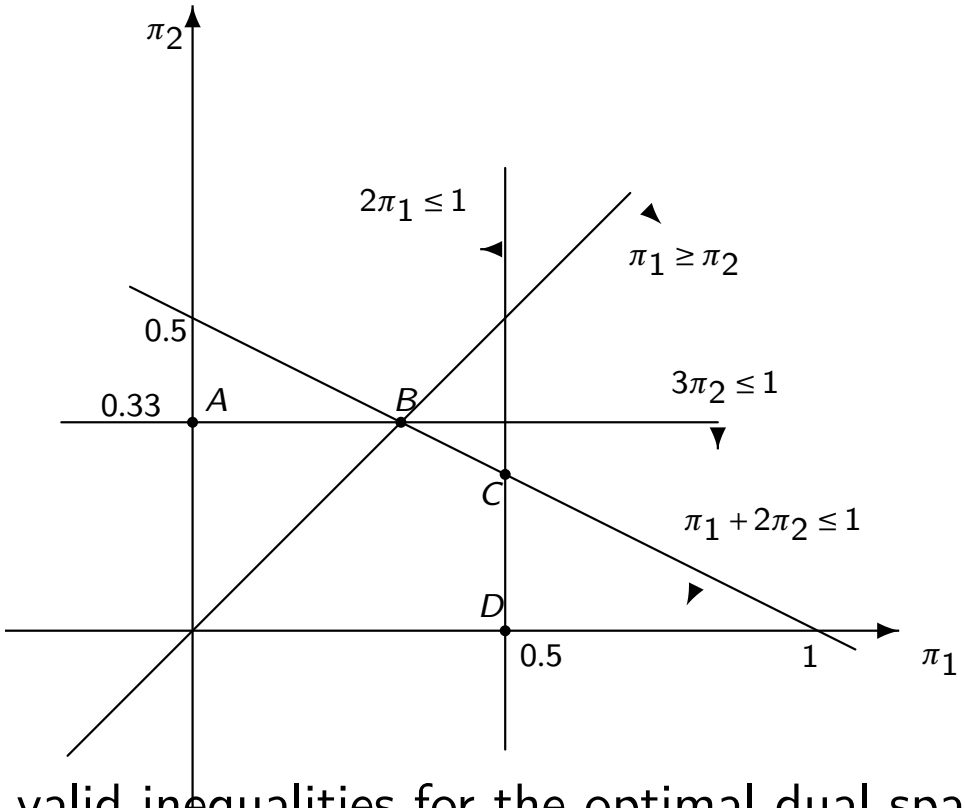
Dual space of CSP: rolls of size 10, items of size 4 and 3



knapsack:

$K = \{(y_1, y_2) : 4y_1 + 3y_2 \leq 10, y_1, y_2 \geq 0 \text{ and integer}\} = \{(2,0),(1,2),(0,3)\}$

Dual space of CSP with cut $\pi_1 \geq \pi_2$



Dual cuts are valid inequalities for the optimal dual space: $\pi_1 \geq \pi_2$ cuts the dual space but obeys all the dual optimal solutions.

Computational implementation of column generation:

- Add dual cuts to model before starting column generation.
- Add starting solution: as suggested by GG, or any other.
- Proceed as usual.

	dual cuts							GG initial solution					
100	d_1	d_2	d_3	d_4	d_5	d_6	d_7	x_1	x_2	x_3	x_4	x_5	
25	-1				-1			4					$\geq d_{25}$
10	1	-1			1	-1			10				$\geq d_{10}$
6		1	-1		1	1	-1			16			$\geq d_6$
3			1	-1		1	1				33		$\geq d_3$
2				1			1					50	$\geq d_2$
min	0	0	0	0	0	0	0	1	1	1	1	1	

Note: every column is a dual constraint.

Computational experiments: instances

Binpacking instances (OR-Library, Beasley'90)

t class: instances with an integer optimum solution in which all bins have three items, which fulfill exactly the capacity of the bin (triplet instances).
Bin capacity is $W = 100$, and item sizes vary between 25.0 and 49.9.
No dual cuts of Type II, because no item can be divided into two smaller items.
Larger instances were tested: the $t501$–instances, with 501 items.

Cutting stock instances (as in Vance'93)

Rolls with widths of 100, 120 or 150, a number of items equal to 200 or 500, with randomly generated real values drawn from a uniform distribution u(1,100).
Existence of small items leads to an explosion in the number of feasible columns.
The more difficult instances are those with larger roll widths and larger number of items, because they have more feasible columns.

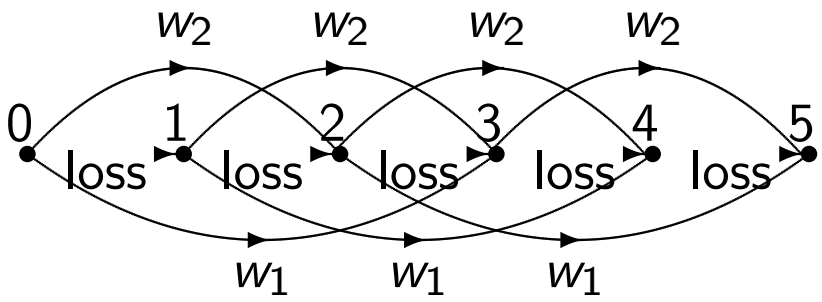
Computational results: summary

Binpacking instances (OR-Library, Beasley'90)
Reduction in number of columns: 43.0 % (from 263.3 to 150.0).
Reduction in computational time: 20.1 % faster.
Reduction in degenerate pivots: percentage falls from 9.3% to 5.4%.

Cutting stock instances (as in Vance'93)
Reduction in number of columns: 75.9 % (from 5309.1 to 1281.6).
Reduction in computational time: 78.2 % (4.5 times faster).
Reduction in degenerate pivots: percentage falls from 39.8% to 8.5%.

Dual cuts in the arc-flow model

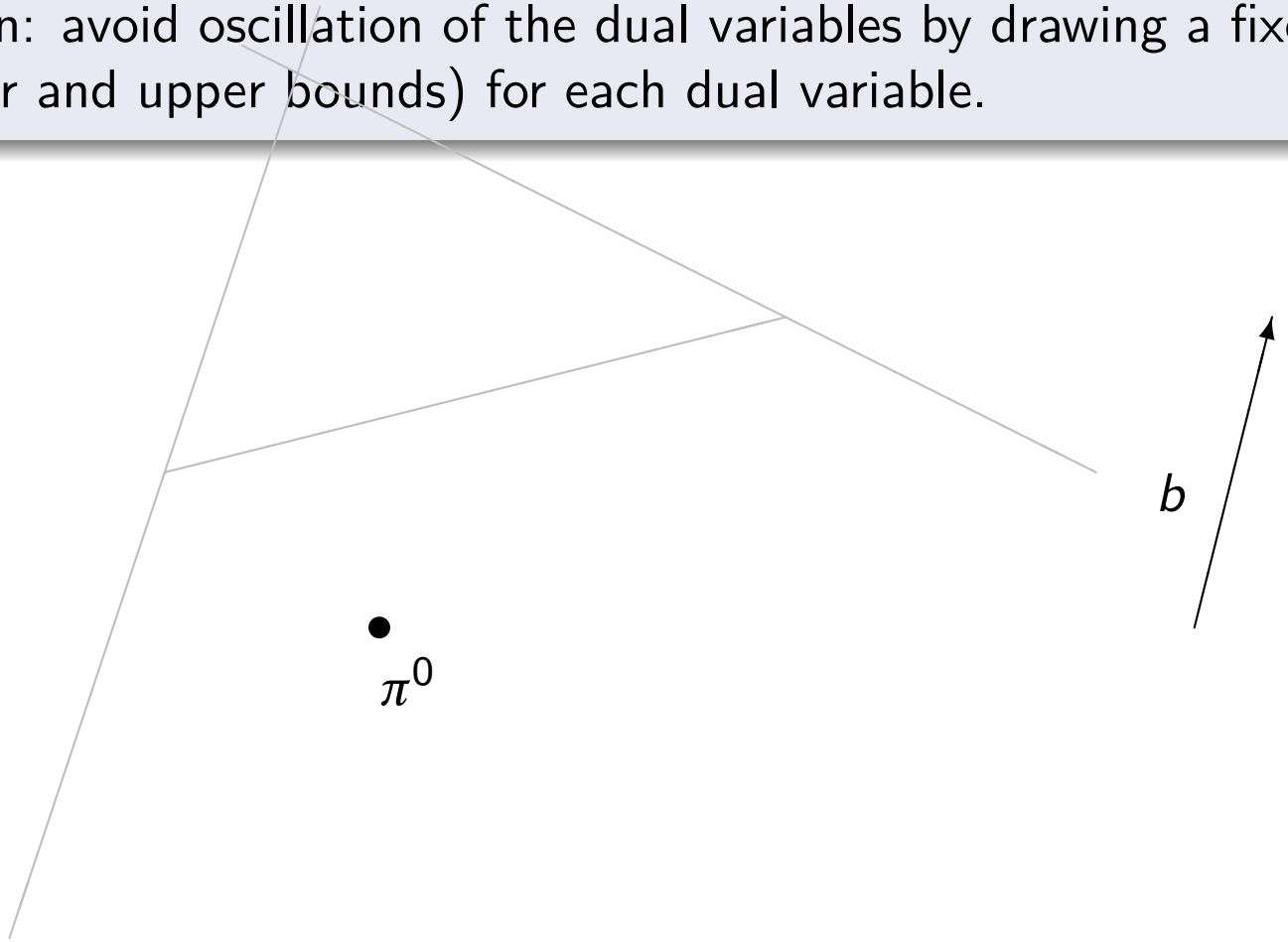
Dual cuts are cycles in the space of the original variables.



Exactly one arc (the largest) is traversed in the direction opposite to its orientation.
Combining a cycle and a path produces a new path.
For each arc with negative flow (direction opposite to its orientation), there is always one (or plus) arc(s) with positive flow(s) with larger value: the net sum of flows in arcs that correspond to a given width is positive (equal to the demand).

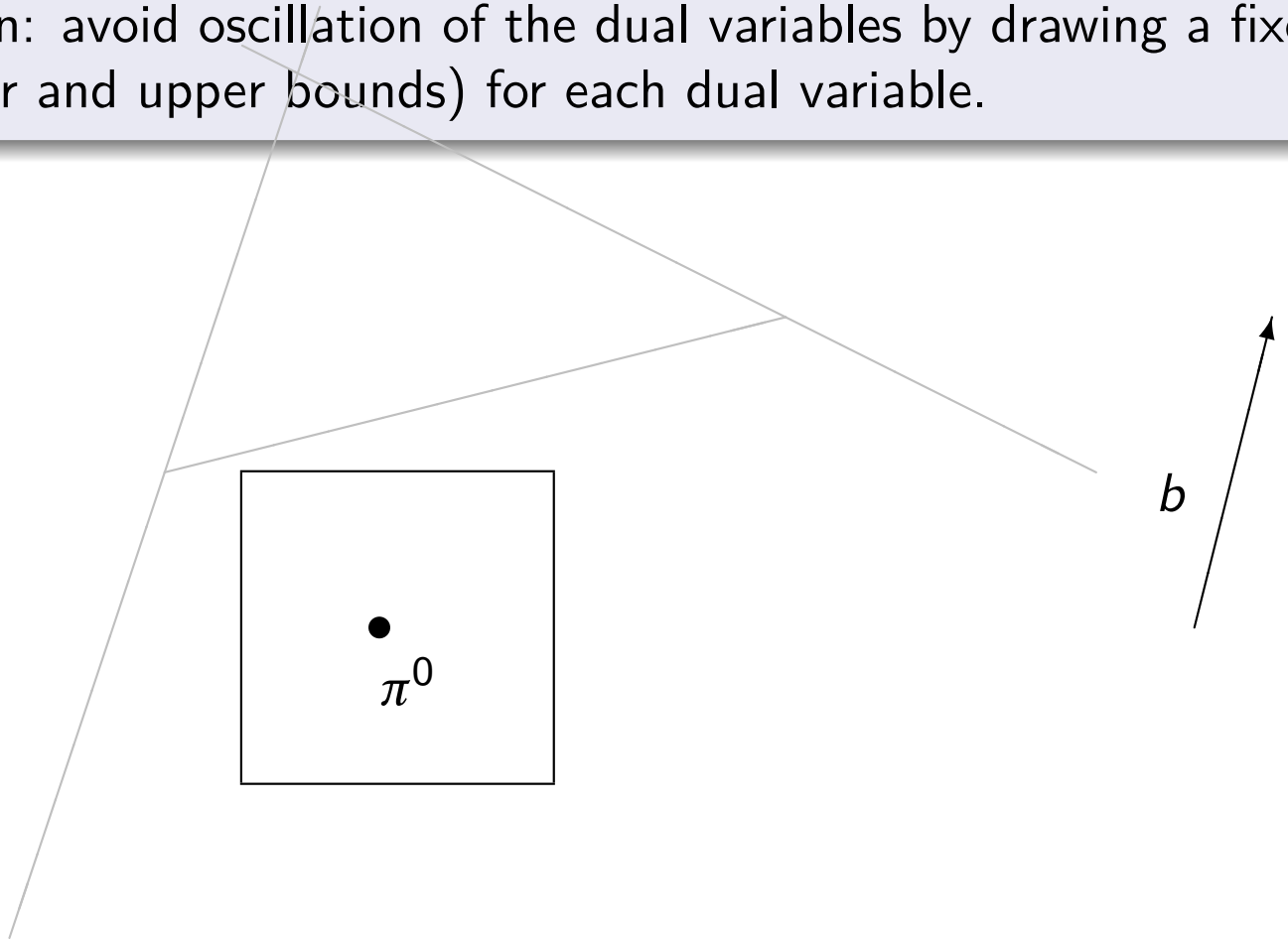
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



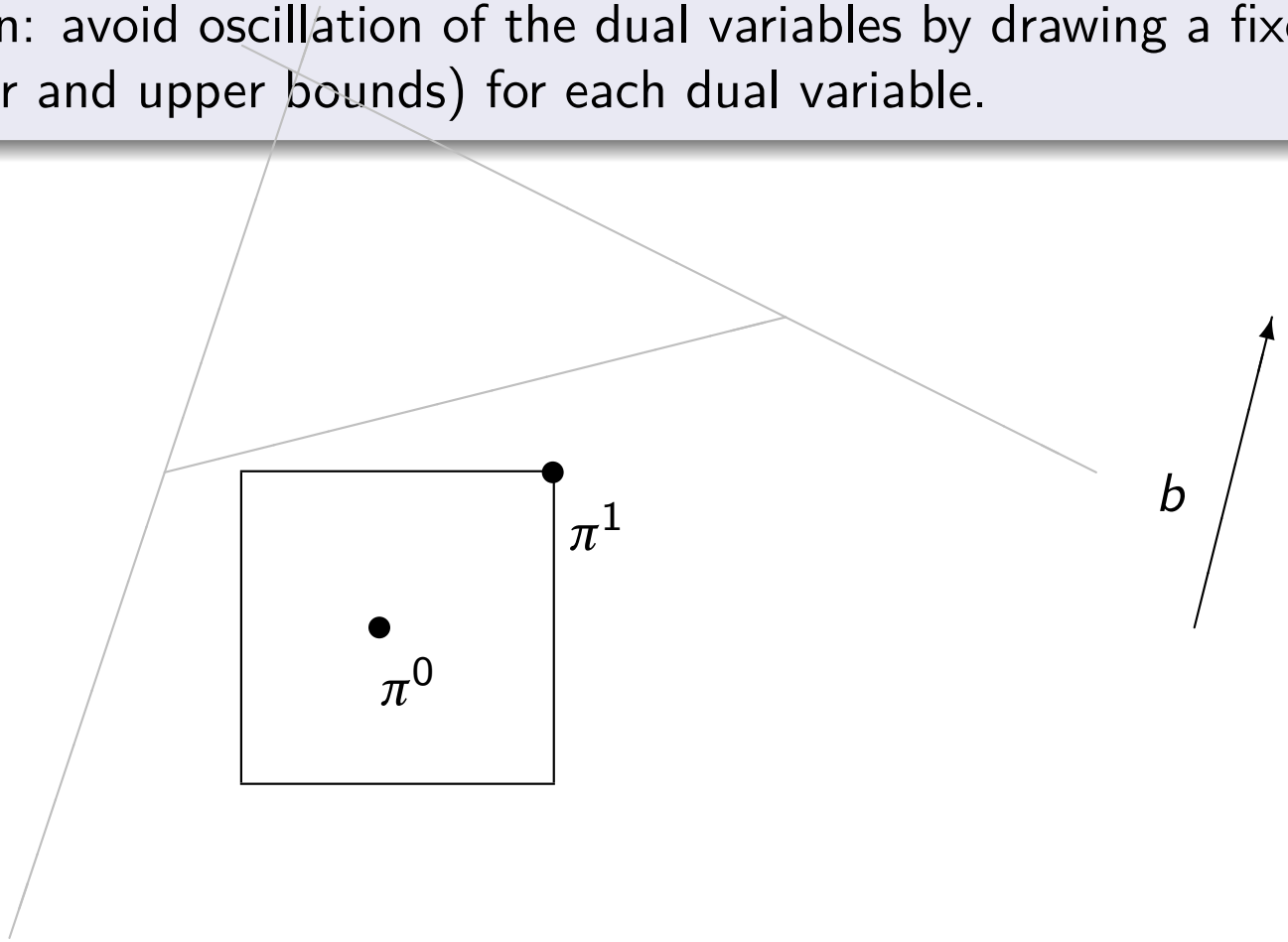
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



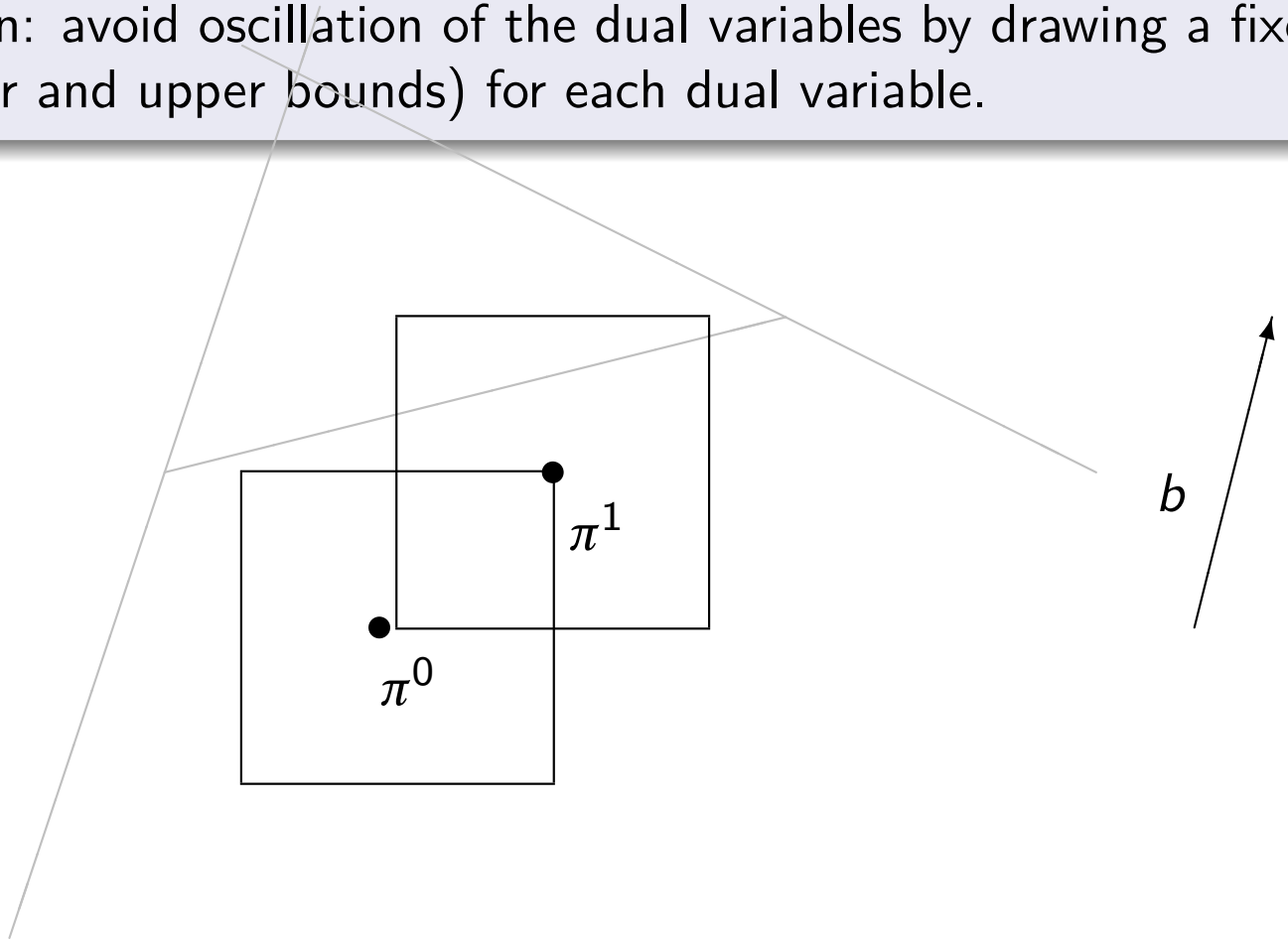
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



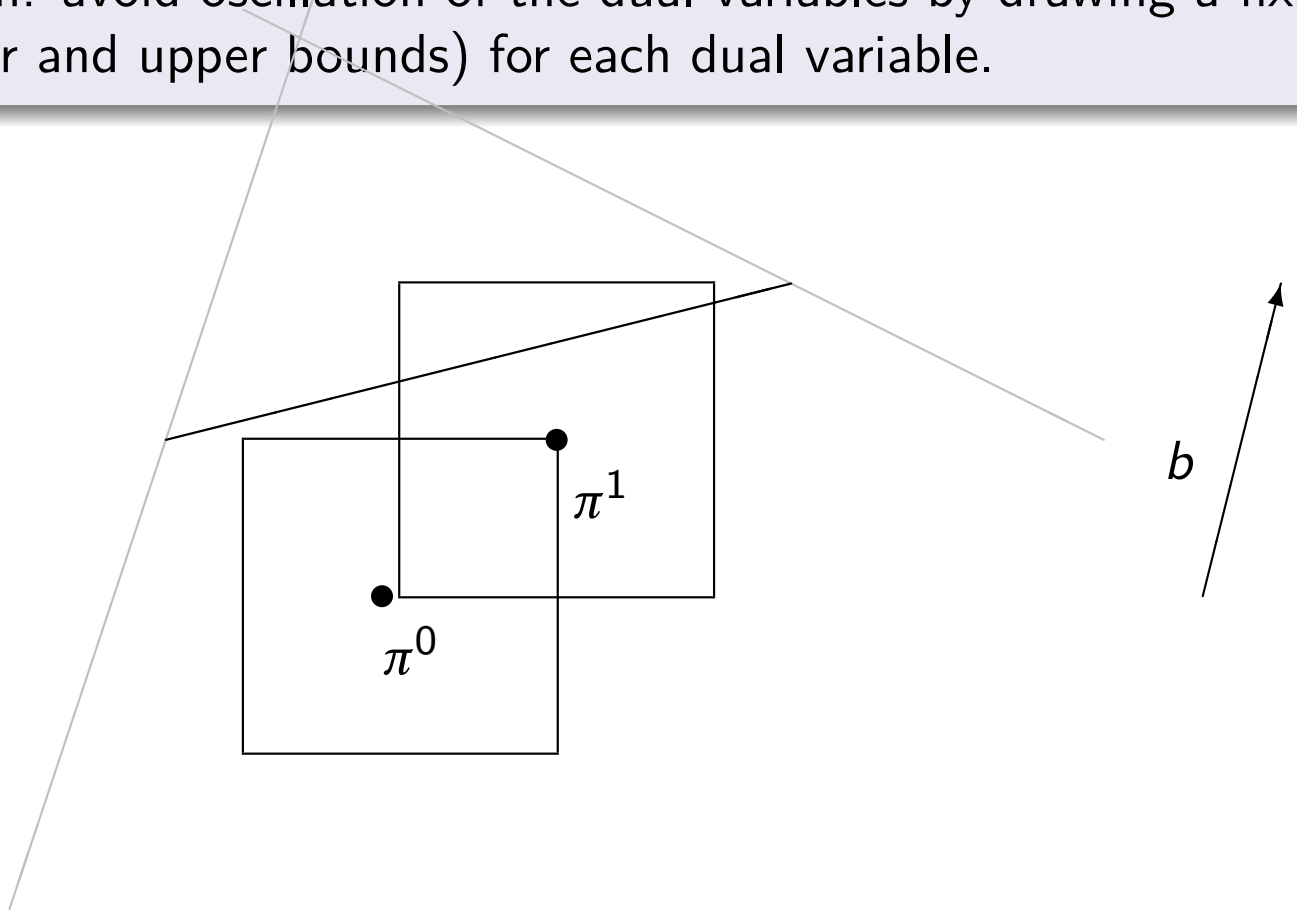
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



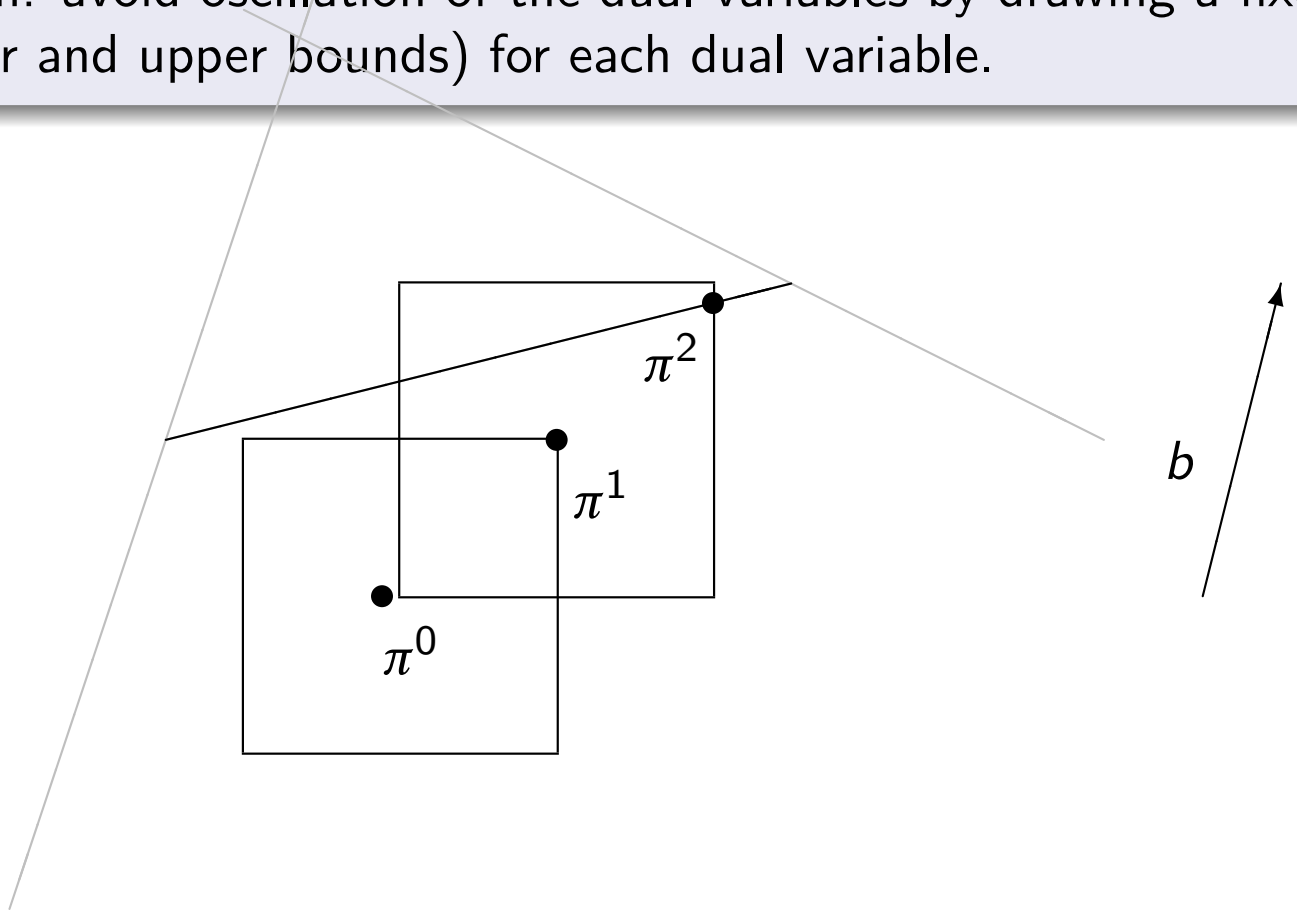
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



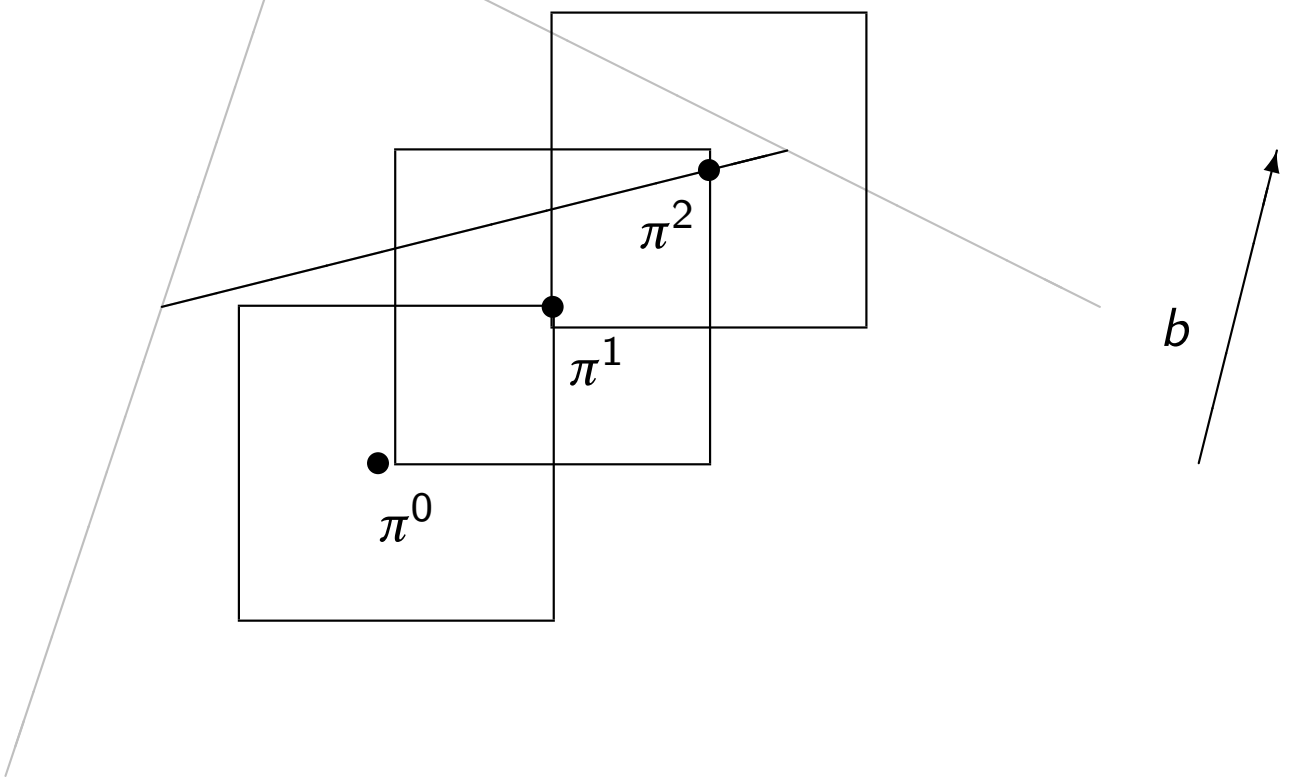
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



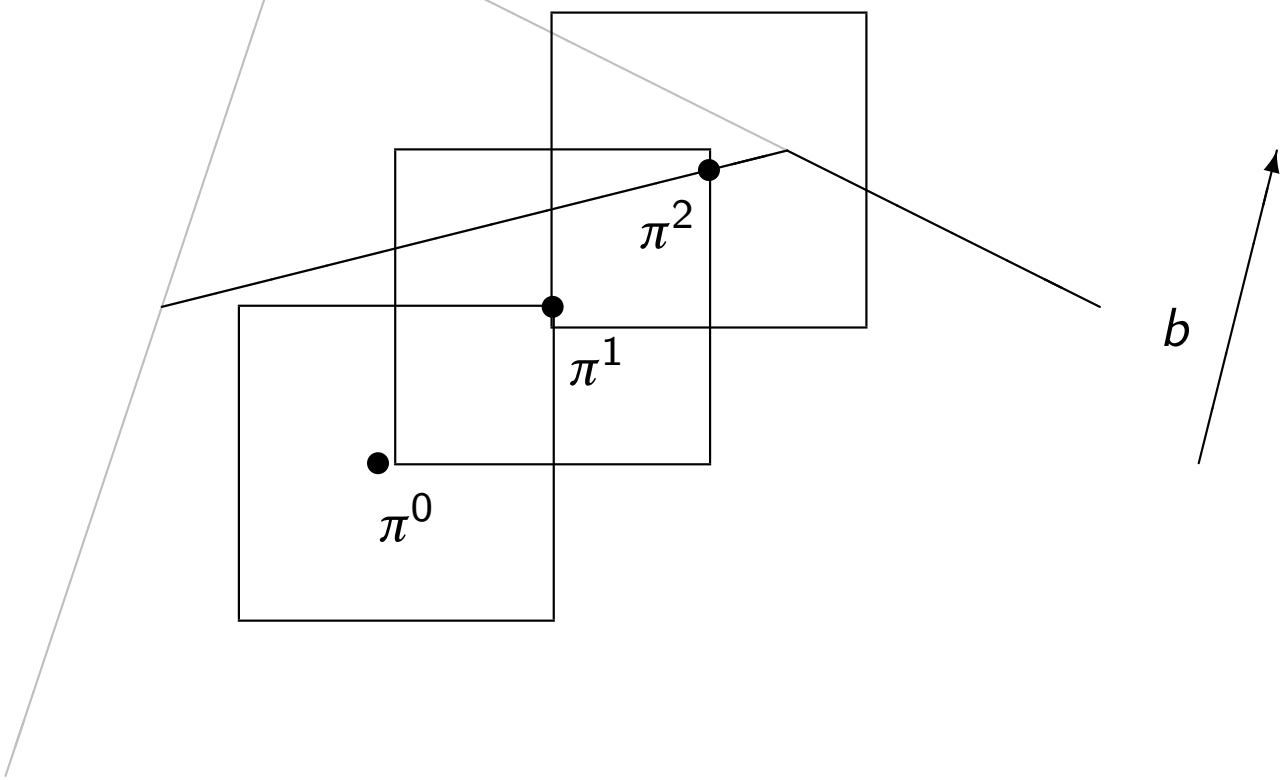
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



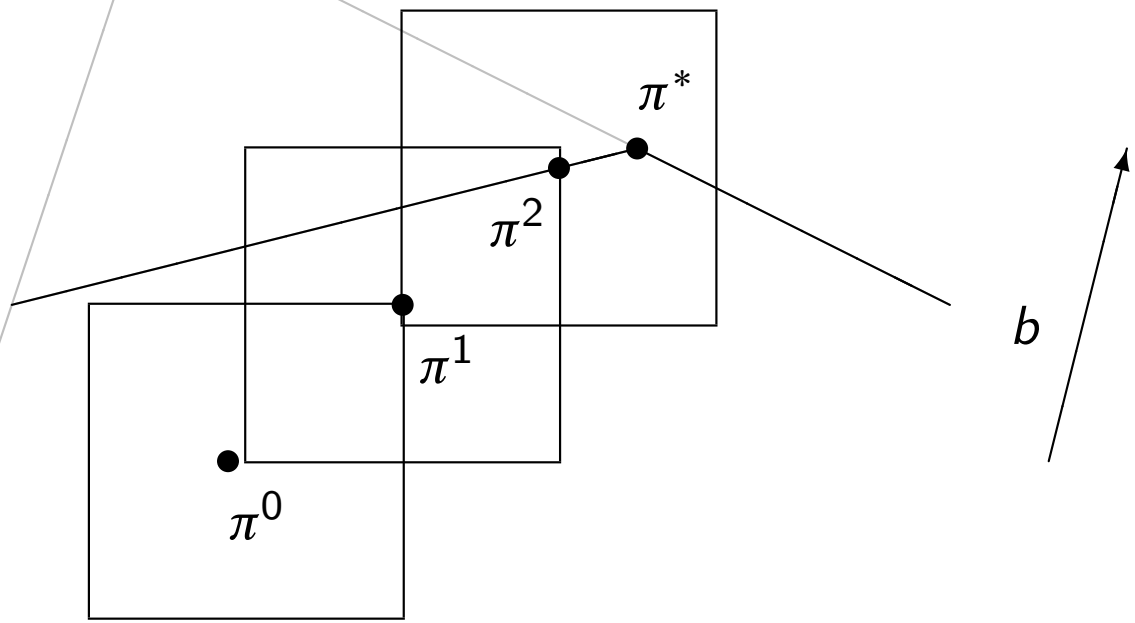
Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



Boxstep method (Marsten et al. 1975)

Motivation: avoid oscillation of the dual variables by drawing a fixed-size Box (lower and upper bounds) for each dual variable.



Solution process:

- If the optimal dual solution is strictly inside the box, then it is an optimal solution to the original problem.
- If any dual variable lies in the boundary of the box (its value equals the lower or the upper bound), the box is re-centered for the next iteration.

Primal and dual problems (Marsten et al. 1975)

$Ax = b$ is original column generation model:

$$(P) \quad \begin{array}{ll} \min & cx \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad (D) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \end{array}$$

Modified column generation model:

$$(P^b) \quad \begin{array}{ll} \min & cx - \delta^- u^- + \delta^+ u^+ \\ s.t. & Ax - u^- + u^+ = b \\ & x, u^-, u^+ \geq 0 \end{array} \qquad (D^b) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \\ & -\pi \leq -\delta^- \\ & \pi \leq \delta^+ \end{array}$$

Bounds in dual variables define a Box: $\delta^- \leq \quad \pi \quad \leq \delta^+$.

Analysis of Boxstep method (Marsten et al. 1975)

Computational implementation of column generation:

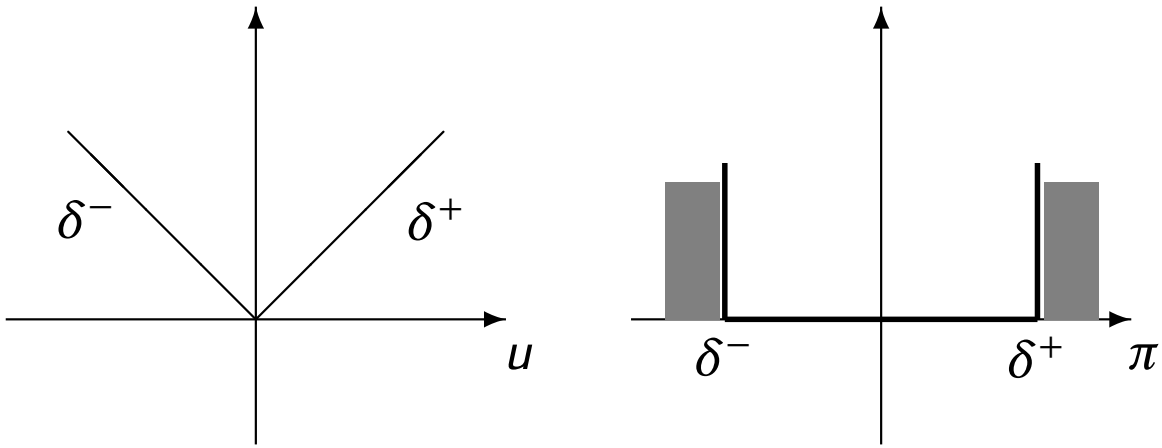
similar to standard, but the restricted master problems has slack and surplus variables with a cost (penalty).

$$(P^b) \quad \begin{array}{ll} \min & cx - \delta^- u^- + \delta^+ u^+ \\ s.t. & Ax - u^- + u^+ = b \\ & x, u^-, u^+ \geq 0 \end{array} \qquad (D^b) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \\ & -\pi \leq -\delta^- \\ & \pi \leq \delta^+ \end{array}$$

- Primal view: penalize deviation from valid solution.
- Dual view: Set Box (Trust region) for dual variables.

- small Box may lead to many iterations.
- standard column generation is Boxstep method with infinite dimension box.

Stabilizing terms: penalty / trust region



- larger penalty in primal, wider box in dual
- smaller penalty in primal, thinner box in dual

Stabilization (du Merle *et al.* 99)

$Ax = b$ is original column generation model:

$$(P) \quad \begin{array}{ll} \min & cx \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad (D) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \end{array}$$

Stabilized column generation model:

$$(P^s) \quad \begin{array}{ll} \min & cx - \delta^- u^- + \delta^+ u^+ \\ s.t. & Ax - u^- + u^+ = b \\ & u^- \leq \epsilon^- \\ & u^+ \leq \epsilon^+ \\ & x, u^-, u^+ \geq 0 \end{array} \qquad (D^s) \quad \begin{array}{ll} \max & \pi b - \epsilon^- w^- - \epsilon^+ w^+ \\ s.t. & \pi A \leq c \\ & -\pi - w^- \leq -\delta^- \\ & \pi - w^+ \leq \delta^+ \\ & w^-, w^+ \geq 0 \end{array}$$

Dual variables may be outside the Box: $\delta^- \leq \pi \leq \delta^+$, but there is a penalty.

Analysis of Stabilization method (du Merle *et al.* 99)

Computational implementation of column generation:

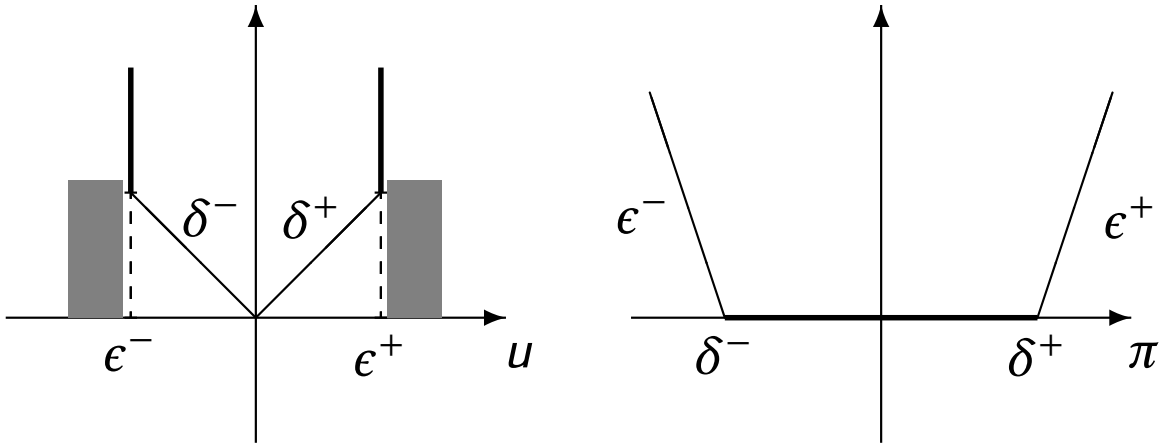
now, the slack and surplus variables with a cost also have bounds.

$$
(P^s)\quad
\begin{array}{lll}
\min & cx-\delta^- u^- +\delta^+ u^+ \\
s.t. & Ax-u^- +u^+ = b \\
 & u^- \le \epsilon^- \\
 & u^+ \le \epsilon^+ \\
 & x,u^-,u^+ \ge 0
\end{array}
\qquad
(D^s)\quad
\begin{array}{lll}
\max & \pi b-\epsilon^- w^- -\epsilon^+ w^+ \\
s.t. & \pi A \le c \\
 & -\pi - w^- \le -\delta^- \\
 & \pi - w^+ \le \delta^+ \\
 & w^-, w^+ \ge 0
\end{array}
$$

- Primal view: penalize deviation from valid solution (now deviation is limited).
- Last two groups of dual constraints: $\delta^- - w^- \le \quad \pi \quad \le \delta^+ + w^+$.
- Dual view: dual variables outside a pre-defined box are penalized (if π is outside the interval $[\delta^-, \delta^+]$, the variables w^- or w^+ take a positive value, penalizing the objective function).

- size of Box is not so critical, because solutions outside Box are allowed.

Stabilizing terms: penalty / trust region



Getting to the optimal solution

Adjustment of penalties at a given restricted master problem:

- If any π is on the border of the interval, the penalty is not sufficiently large, and the optimal solution of the stabilized problem may not be valid for the original problem.
- Adjust penalties !
- The algorithm needs appropriate strategies for the adjustment of the penalties so that the optimal solution is found rapidly.

At the optimal solution:

- Complementary Slackness Theorem: if the optimal value of π is strictly inside the interval $[\delta^-, \delta^+]$, the constraints have slack and the corresponding dual variables are null, that is, $u^- = u^+ = 0$, which implies that $Ax = b$ (valid for original model).
- The same happens with $\epsilon^- = \epsilon^+ = 0$.

Dual-optimal inequalities and deep dual-optimal inequalities

Instead of just referring to "dual cuts", at some points, we will make a distinction between the two different classes:

Dual-optimal inequalities:
All dual optimal solutions are preserved (as in the dual cuts for Gilmore-Gomory model for the CSP).

Deep dual-optimal inequalities:
One may even effectively cut a subset of dual optimal solutions, if, at least, one dual optimal solution is preserved.
One optimal dual solution is sufficient to drive the process to find the optimum.

Perturbing dual-optimal inequalities

Consider now the set of dual-optimal inequalities perturbed by an ε: $\pi D \leq d + \varepsilon$:

$$(P^{e'}) \quad \begin{array}{ll} \min & cx + (d+\varepsilon)y \\ s.t. & Ax + Dy = b \\ & x, y \geq 0 \end{array} \qquad (D^{e'}) \quad \begin{array}{ll} \max & \pi b \\ s.t. & \pi A \leq c \\ & \pi D \leq d + \varepsilon \end{array}$$

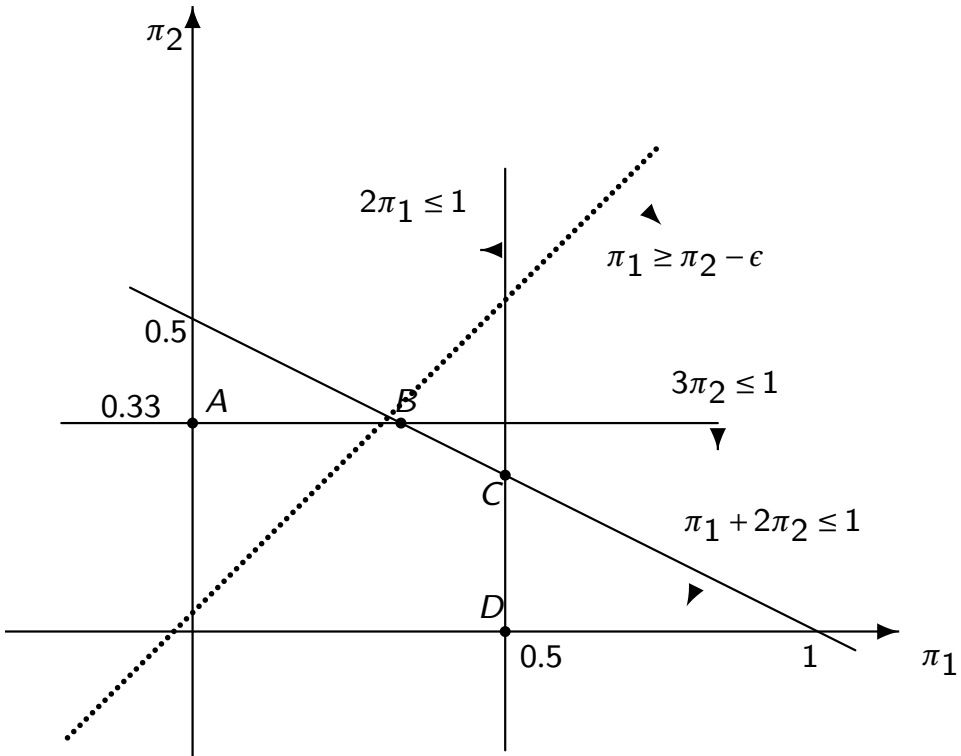
Proposition

Let $\pi D \leq d$ be a set of dual-optimal inequalities, and (x^*, y^*) an optimal solution for $P^{e'}$. Then, $y^* = 0$ and x^* is an optimal solution for P.

Proof: All dual-optimal solutions obey $\pi D \leq d$ and have slack in $\pi D \leq d + \varepsilon$. By complementary slackness, the corresponding primal variables $y^* = 0$. □

(Ben Amor,Desrosiers,VC'2006)

Dual space of CSP with cut $\pi_1 \geq \pi_2$ perturbed by ϵ



Columns of dual cuts will be 0 in any optimal solution [Ben Amor, Desrosiers, VC, 2006].

Dual inequalities for MLCSP

Dual-optimal inequality:

$$v_k - v_{k'} \leq W_k - W_{k'}, \quad k = 1, ..., K-1, k' = 2, \ldots, K, \ W_k > W_{k'}.$$

Example: $v_1 - v_2 \leq 7 - 5$.
Packing pattern of a bin can always be reassigned to a bigger bin at a cost equal to the difference between their capacities (similar to cuts for CSP, but used for bins).

Deep dual-optimal inequality:

$$v_K \geq W_K - W_1$$

Example: $v_3 \geq 3 - 7$.

Proofs omitted.
Both inequalities can be perturbed by ε to force corresponding primal variables to 0.

Deep dual-optimal inequalities for MLCSP: example

$v_3 \geq -3-\varepsilon$ is a deep dual-optimal inequality (not in the tableau): sol.1 is cut, but alternative optimal dual solution sol.2 is preserved.

	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2			dual sol.1	dual sol.2
w_i =3	2		1		1				≥	5	24	3
2		3	1	2		2			≥	6	16	2
1	1	1		1	1				≥	4	8	1
W_k =7	1	1					1		≤	2	-49	0
5			1	1			-1	1	≤	1	-35	0
4					1	1		-1	≤	3	-28	0
min	7	7	5	5	4	4	2	1				
primal	2.0	0.0	0.0	1.0	1.0	2.0	0.0	0.0			z^* = 31.0	31.0

Optimal solution: all bins are used: $\sum_{i=1}^{m} w_i b_i = \sum_{k=1}^{K} W_k b_k = 31$.
In this case, $((k+1)w_1, (k+1)w_2, (k+1)w_3, -kW_1, -kW_2, -kW_3)$ is an optimal dual solution for $k \geq 0$.

When an optimal dual solution is known

Given an optimal dual solution $\overline{\pi}^*$ for $\overline{D}$ and a vector of scalars $\Delta > \mathbf{0} \in \mathbb{R}^m$, use the stabilized pair of primal and dual problems:

$$
\begin{array}{c|c}
v(\overline{P}(\overline{\pi}^*)) := \min \mathbf{c}^T\mathbf{x} - (\overline{\pi}^* - \Delta)^T\mathbf{y}_1 + (\overline{\pi}^* + \Delta)^T\mathbf{y}_2 & v(\overline{D}(\overline{\pi}^*)) := \max \mathbf{b}^T\pi \\
A\mathbf{x} - \mathbf{y}_1 + \mathbf{y}_2 = \mathbf{b} & A^T\pi \leq \mathbf{c} \\
\mathbf{x} \geq \mathbf{0},\ \mathbf{y}_1 \geq \mathbf{0},\ \mathbf{y}_2 \geq \mathbf{0} & \overline{\pi}^* - \Delta \leq \pi \leq \overline{\pi}^* + \Delta.
\end{array}
$$

Proposition

Let $E^T\pi \leq \mathbf{e}$ be a set of deep dual-optimal inequalities and $(\overline{\mathbf{x}}^*, \mathbf{y}_1^*, \mathbf{y}_2^*)$ be an optimal solution for $\overline{P}(\overline{\pi}^*)$. Then, $\mathbf{y}_1^* = \mathbf{y}_2^* = \mathbf{0}$ and $\overline{\mathbf{x}}^*$ is an optimal solution for P.

Proof: This result was proved in a previous session. □

When an optimal dual solution is known: example

Proposition

Consider a CSP instance with no loss at optimality. Then, $\pi_i^* = \frac{w_i}{W}, i \in I$ is an optimal dual solution.

Proof: All dual constraints are obeyed. The dual objective function reaches the optimal value $\sum_{i=1}^{m} b_i w_i / W$. Therefore, this dual solution is optimal. □

Computational results for binpacking triplet instances (OR-Library, Beasley'90)

- Reduction in number of columns: 90.2 % (from 124.2 to 12.2).
- Size of box $\Delta = 10^{-2}$

(Ben Amor,Desrosiers,VC'2006)

Complicating issues in the branch-and-price tree

Cuts were derived for the solution of the LP relaxation.
Some dual cuts fornot be valid in the branch-and-price tree.
Keep the valid cuts
Derive new cuts
Do not relax the optimal solution of a branch-and-price node.

First, let us show that some dual cuts are not valid in the branch-and-price tree:

Example: optimal solution of root node

Stock: lengths $W = (7,5)$ with availabilities $B = (5,5)$
Items: sizes $w = (4,3,2)$ and demands $b = (2,5,2)$

	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	λ_{21}	λ_{22}	λ_{23}		
w_i =4	1	1				1			$\geq$	2
3	1		2	1			1		$\geq$	5
2		1		2	3		1	2	$\geq$	2
W_k =7	1	1	1	1	1				$\leq$	5
5						1	1	1	$\leq$	5
	7	7	7	7	7	5	5	5		

primal	2.0	0.0	0.5	0.0	0.0	0.0	2.0	0.0	z^* =	27.5

Arc-flow model solution: $x_{0,4} = 2$, $x_{4,7} = 2$, $x_{0,3} = 2.5$, $x_{3,6} = 0.5$ and $x_{3,5} = 2$.

Example: optimal solution of a node of the branch-and-price tree

Optimal solution after branching constraint $x_{0,3} \leq 0$ on arc $(0,3)$ is enforced:

	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	λ_{21}	λ_{22}	λ_{23}	
w_i =4	1	1				1			≥ 2
3	1		2	1			1		≥ 5
2		1		2	3		1	2	≥ 2
W_k =7	1	1	1	1	1				≤ 5
5						1	1	1	≤ 5
$x_{0,3} \leq 0$			1	1			1		≤ 0
	7	7	7	7	7	5	5	5	

primal | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | $z^* =$ 40.0

This branching constraint is just for illustration purposes (it would not be selected in the branching scheme at this point).

Example: dual inequality relaxes model

Optimal solution after branching constraint $x_{0,3} \leq 0$ on arc $(0,3)$ is enforced, and there is a dual cut, corresponding to d_1 :

	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	λ_{21}	λ_{22}	λ_{23}	d_1	
w_i =4	1	1				1			-1	$\geq$ 2
3	1		2	1			1		1	$\geq$ 5
2		1		2	3		1	2		$\geq$ 2
W_k =7	1	1	1	1	1					$\leq$ 5
5						1	1	1		$\leq$ 5
$x_{0,3} \leq 0$			1	1			1			$\leq$ 0
	7	7	7	7	7	5	5	5	0	

primal | 3.5 | 0.0 | 0.0 | 0.0 | 2/3 | 0.0 | 0.0 | 0.0 | 1.5 | z^* = 29 1/6

Extra column replaces an item with size 4 by another with size 3, in the pattern associated to λ_{11}.
The resulting pattern is the same as the one associated to λ_{13}.
There is a positive flow in arc $(0,3)$, to which the branching constraint does not apply.

What causes the trouble...

Dual cuts are cycles in terms of the arc-flow model.
There are cycles involving the same exchanges but starting at different positions.
They are mapped into the same dual cut of Gilmore-Gomory model, when applying the Dantzig-Wolfe decomposition, but branching constraints act on single arcs.

When, combining a pattern P_i with a column related to a dual cut results in a pattern P_j with one of the following characteristics

- pattern P_j has a null coefficient in the branching constraint on an arc (s,t), when P_j does in fact translate into a set of arcs that include this arc (as in the last Example).
- pattern P_j has a $+1$ coefficient in the row of a branching constraint enforced on an arc (s,t), but P_j does not translate into a set of arcs that include (s,t) (as in the next Example);

the branching constraint enforced on an arc (s,t) does not act correctly over the columns producing flow on (s,t).

Another example: dual inequality causes trouble

Optimal solution after branching constraint $x_{0,3} \leq 0$ on arc $(0,3)$ is enforced, and there is a dual cut, corresponding to d_2 :

	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	λ_{21}	λ_{22}	λ_{23}	d_2	
w_i =4	1	1				1				≥ 2
3	1		2	1			1		-1	≥ 5
2		1		2	3		1	2	1	≥ 2
W_k =7	1	1	1	1	1					≤ 5
5						1	1	1		≤ 5
$x_{0,3} \leq 0$			1	1			1			≤ 0
	7	7	7	7	7	5	5	5	0	

Combining column λ_{14} with column d_2 yields a new column, similar to λ_{15} but with a $+1$ coefficient in the row of a branching constraint enforced on arc $(0,3)$, but the new column does not have arc $(0,3)$.

Two sufficient conditions for validity

A subset of dual inequalities for the CSP remain valid in node w of the branching tree.
Subset depends on the specific set of arcs on which a branching constraint is enforced.

Proposition

For all the arcs (s,t) on which at least one branching constraint has been enforced at node w, and for an item i and a subset S of the item sizes such that $w_i \geq \sum_{l \in S} w_l$, if at least one of the following conditions holds

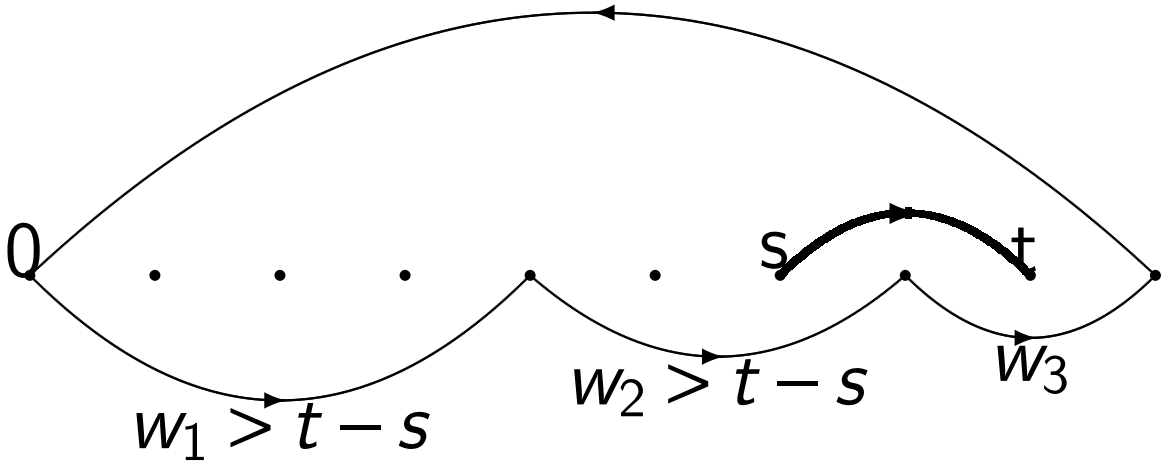
- $t - s > w_i$: *branching constraints act on items larger than w_i,*
- $\sum_{l \in S : w_l > t-s} w_l > s$: *dual cut will not produce size $(t-s)$ before position s,*

then $u_i \geq \sum_{l \in S} u_l$ will be a valid dual-optimal inequality for D^e at node w.

Condition (1): Branching on larger arcs (typical in branching scheme) enables keeping the dual cuts for smaller items.
Condition (2): see next page

About the validity of condition (2)

Dual cut will not produce size $(t-s)$ before position $s : \sum_{l \in S: w_l > t-s} w_l > s$



There may be arcs of length $(t-s)$ in the dual cut (strictly after position s), but branching constraint on arc (s,t) will not be "cheated". Gilmore-Gomory model assumes items placed in order of decreasing sizes.

New family of dual inequalities

Inequalities depend on the dual variables for the demand and branching constraints.

Proposition

At a node w, and for a single item i and subset of item sizes S such that $w_i \geq \sum_{l \in S} w_l$, the following dual inequalities

$$u_i \geq \sum_{l \in S} u_l + \sum_{\{l \in G^w : t - s \leq w_i\}} \mu^l_{s,t} - \sum_{\{l \in H^w : t - s \leq w_i\}} \nu^l_{s,t}, \quad i = 1, \ldots, m, \ \forall S \quad (10)$$

are valid dual-optimal inequalities for D.

G^w, H^w : sets of branching constraints of the types $\leq$ and $\geq$, respectively in node w.
$\mu^l_{s,t}, \nu^l_{s,t}$: dual variables associated to $\leq$ and $\geq$ constraints, respectively, indexed by l, acting on flow of arc (s,t).

New family of dual inequalities: example

$$u_2 \geq u_3 + \sum_{\{l \in G^w : t-s \leq 3\}} \mu^l_{s,t} - \sum_{\{l \in H^w : t-s \leq 3\}} \nu^l_{s,t}$$

	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	λ_{21}	λ_{22}	λ_{23}	d_2	
w_i =4	1	1				1				≥ 2
3	1		2	1			1		-1	≥ 5
2		1		2	3		1	2	1	≥ 2
W_k =7	1	1	1	1	1					≤ 5
5						1	1	1		≤ 5
$x_{0,4} \geq 1$	1	1				1				≥ 1
$x_{0,3} \geq 1$			1	1			1		-1	≥ 1
$x_{0,3} \leq 4$			1	1			1		1	≤ 4
$x_{3,6} \geq 1$			1						-1	≥ 1
$x_{3,6} \leq 2$			1						1	≤ 2
$x_{3,5} \leq 1$				1			1		1	≤ 1
	7	7	7	7	7	5	5	5	0	

There is a pattern column that dominates a combination of a pattern column with a dual cut column.

Computational results: summary

Results for MLCSP improve significantly when dual inequalities are applied in the whole branch-and-bound tree (stabilized branch-and-price-and-cut algorithm).
Using cuts that remain valid in branching tree and new family of cuts.

50 hard instances (Belov'2002)
Optimum is reached using less 70% branching nodes.
Computing time is reduced 50%.
47 were solved to optimality, instead of 45.

A word about a nice result

Different primal models with equally constrained dual spaces take the same number of iterations.

Experiment 1
BinaryCSP model (disaggregated demand): there is a constraint for each item of the same size (demand is equal to 1).
Solution of BinaryCSP takes more iterations than CSP.
Experiment 2
Add dual constrains saying that dual variables of items of the same size should be equal.
Solution of BinaryCSP takes approximately the same number of iterations as CSP.

(Ben Amor,Desrosiers,VC'2006)

Approaches

Stabilization
General framework, which is not problem dependent.
Adjustments of stability center may be needed.

Dual cuts
Derivation relies on characterization of the space of dual optimal solutions.
Problem dependent, not easy to derive.
Valid through entire column generation process.

Combination
Using dual cuts amounts to solving an alternative primal model (equally strong) with a more restricted dual space.
Stabilization and dual cuts can be combined.

Concluding remarks

- Strength of models is of crucial importance.
- Dual cuts make column generation faster keeping models strong.
- Restriction of dual space may be an important factor for faster convergence.

Part VI

Practical issues, accelerating strategies and heuristics

Outline

- Pre-processing
- Master problem
- Subproblem
- Branch-and-bound

Pre-processing

- Arc elimination
- Initial Primal Solutions

Arc elimination

Using a feasible primal integer solution and a feasible dual solution to the relaxation of the problem to fix a *path variable* to 0:

Lemma

- *Given* (IP) : $\min\{cx : Ax = b, x = (x_i)_{i\in\{1,\ldots,n\}} \in \{0,1\}^n\}$.
- π : *feasible dual solution of the LP-relaxation of IP, with value* πb.
- U : *upper bound for IP, given by a feasible primal integer solution.*
- *If, for some* $p \in \{1,\ldots,n\}$, $\pi b + (c_p - \pi A_p) > U$, *then* $x_p = 0$ *in all optimal solutions of IP.*

- In the integer problem (IP), x_p is a binary variable.
- Idea of proof: the lagrangean lower bound of the problem with $x_p = 1$ is above the upper bound.

Arc elimination (cont.)

Proof:

- Consider modified problem where x_p is fixed to 1 and $x_i \geq 0, \forall i \in \{1,\ldots,n\}\backslash\{p\}$:

$$\min\{cx + c_p : Ax = b - A_p, x_i \geq 0, \forall i \in \{1,\ldots,n\}\backslash\{p\}, x_p = 0\}.$$

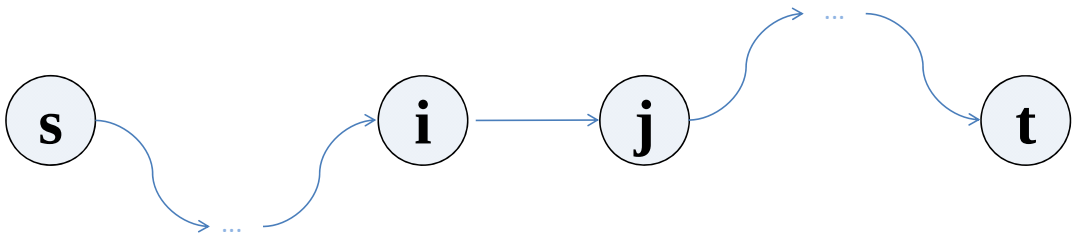
- Dual of modified problem: $\max\{(b - A_p)\pi : \pi A \leq c\}$
- lagrangean of modified problem:

$$\begin{aligned} z_{LR}(\pi) &= c_p + \min_{x\backslash x_p}\{cx + \pi((b - A_p) - Ax)\} = \\ &= (c_p - \pi A_p) + \pi b + \min_{x\backslash x_p}\{(c - \pi A)x)\} \end{aligned}$$

- in dual feasible solution, $\pi A \leq c$.
- given a feasible dual solution π, the lagrangean lower bound of modified problem $> U$.

Arc elimination (cont.)

Using a feasible primal integer solution and a feasible dual solution to the relaxation of the problem to fix an *arc variable* to 0:



Variable fixing

- Consider the paths $p = (s, \ldots, i, j, \ldots, t)$ that contain arc (i, j).
- If $\pi b + \min_{p:(i,j)\in p}(c_p - \pi A_p) > U$, then arc (i, j) can be eliminated.

Initial Primal solutions

Starting solutions for column generation:

- Polynomial heuristics: First Fit Decreasing and Best Fit Decreasing
- Pseudo-polynomial heuristics

First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) heuristics

FFD : largest unplaced item is assigned to the bin with smallest index already used with sufficient remaining capacity; if there is none, a new bin is started.
BFD : largest unplaced item is assigned to the bin with smallest remaining capacity, but still sufficient to accommodate the item; if there is none, a new bin is started.
FFD and BFD have absolute performance ratios of $3/2$, *i.e.*, $z_H \leq 3/2\ z^*$, where z^* is the value of the optimum (Simchi-Levi'94).
Absolute performance ratios:

First-Fit Decreasing – $\frac{3}{2}$.

Best-Fit Decreasing – $\frac{3}{2}$.

Asymptotic performance ratios:

First-Fit Decreasing – $\frac{11}{9}$.

Best-Fit Decreasing – $\frac{11}{9}$.

FFD: example of asymptotic performance ratio of 11/9

Asymptotic performance ratio: happens even in large instances.
Example: optimal solution uses $9N$ bins, heuristic solution uses $11N$ bins.

51		26	23
51		26	23
51		26	23
51		26	23
51		26	23
51		26	23
27	27	23	23
27	27	23	23
27	27	23	23

$z^* = 9N$

51		27		
51		27		
51		27		
51		27		
51		27		
51		27		
26	26	26		
26	26	26		
23	23	23	23	
23	23	23	23	
23	23	23	23	

$z_{FFD} = 11N$

Absolute performance ratio: only in instances with a small number of bins.

Pseudo-polynomial heuristics

Greedy (myopic) heuristic, based on iterative solution of knapsack problems:

Build list with all items
While (there are items in the list) do
    solve knapsack problem
    remove items in the solution from the list
    (repeat removal, if there are multiple copies of all items)
End While

- Computation time is not significant in the column generation framework.
- Usually provides good starting solutions, with, at least, some very good cutting patterns.
- Last patterns may be very poor.

Pseudo-polynomial heuristics (two implementations)

Vanderbeck'99:

- among the solutions with maximum capacity usage, $\sum_i w_i y_i$, choose the one that is lexicographically smaller when considering a solution vector where the items are ordered by non-increasing sizes $(w_1 \geq w_2 \geq \ldots \geq w_m)$.

VC'05:

- use weights w_i = item sizes and profits $p_i = (1-(j-1)/n)\ w_i, \forall i$, to favor choice of solutions with larger items, leaving the smaller items, which should be easier to combine, to subsequent iterations
- preferable to solving a subset-sum problem, $\max\{\sum_i w_i y_i : \sum_i w_i y_i \leq W, y_i \geq 0 \text{ and integer}, \forall i\}$, which is, in practice, difficult to solve [Martello, Toth'90].

They provide much better starting solutions than FFD or BFD.

Master Problem

- Column elimination
- Constraint aggregation
- Multiple columns at each iteration
- Stabilization [Part V]
- Dual cuts (including covering vs. partitioning constraints) [Part V]

Column elimination

Most probably many columns of the RMP will not be used in the optimal solution. Periodically,

- purge columns with reduced cost above a pre-defined threshold, or
- purge columns with zero value for a predefined number of iterations.

Constraint aggregation

CSP [Alves, VC'07]

- Basic algorithm:
  - pick two similar item sizes and aggregate demands (*e.g.*, use larger item size).
  - less constraints and smaller subproblem
  - solution of aggregated model may be "sub-optimal"
  - at the end, disaggregate to check if re-optimization is necessary.
- Also more effective n-phase algorithm.

VR & CS [Elhallaoui, Villeneuve, Soumis, Desaulniers'05]

- Vehicle routing and crew scheduling:
- Aggregation according to an equivalence relation that changes dynamically over time.
- Shortest path problem used to recover the non-aggregated dual information
- Master problem time reduced by a factor of 8.

Multiple columns at each iteration

- when problem has several subproblems, use the dual information of the RMP to generate columns from all the subproblems, and insert them all in the RMP, before re-optimizing.
- if possible, pick not only the optimal solution of subproblem, but also $2^{nd}, \ldots, k^{th}$ best solutions (there are cases with 3 to 10 columns).
- furthermore, use heuristics to find columns that are orthogonal together with the most attractive (*i.e.*, that may combine better to form a solution).

Subproblem

- Heuristic pricing
- State space reduction

Heuristic pricing

- Subproblem may be a problem difficult to solve practically.
- Try to get close to the optimal solution using heuristics.
- Only resort to solving subproblem optimally when no more attractive solutions are found.

Example

- use different heuristic algorithms along the process.

Space state reduction

- Temporarily reduce the burden of the dynamic programming subproblem,
- Then resort to solving subproblem optimally.

Example

- in problems with time constraints, start with a less precise definition of time,
- start with a subset of clients or items.

Branch-and-bound

- Early branching
- Upper bounds

Early branching

- For problems with integer cost coefficients, c_j, $\forall j$, given
- $\overline{z}$: value of the current solution of column generation process, and
- LB : a lower bound,
- if $\lceil \overline{z} \rceil = \lceil LB \rceil$,
- column-generation process can be cut off to reduce the tail.

- One can also terminate heuristically earlier.

Upper bounds

- use depth first search (possibly making a single dive) to try to get a good incumbent solution, which may help fathoming nodes later during the full exploitation of the tree.
- use heuristics (more or less elaborate) at each node of the tree.

Other issues

- G. Desaulniers, J. Desrosiers, M. Salomon, Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems, Cahiers GERAD G-99-36, and in Essays and Surveys in Metaheuristics, C. Ribeiro and P. Hansen (eds.), Kluwer Academic Publishers, 309-324, 2002.

Concluding remarks

- Practical issues may reduce computational time significantly.
- They are problem dependent, and have to be tailored.

Primal cutting planes

Outline

- Separation and row generation
- Row generation in column generation models: compatibility
- Strengthening column generation models
- Super-additive Non-decreasing Functions and Dual Feasible Functions
- Application: multicommodity flow problem
- Application: minimization of number of set-ups in CSP

Separation problem

Given a fractional solution, the *separation problem* finds a valid inequality for the integer problem that cuts the current fractional solution.

Given a family of valid inequalities, we may (try to) search the valid inequality (primal cut) of the family which is more violated.

Column generation models with primal cuts: compatibility

- Primal cuts lead to stronger models.
- When primal cut is explicitly inserted in the RMP,
- and provides dual information,
- in the subproblem, we must be able to anticipate the coefficient of the column in the primal cut,
- so that the attractiveness of the column is correctly evaluated.

Valid inequalities

should be easy to deal with if they can be expressed in terms of the original variables:

- add constraint to RMP,
- use dual information of constraint to change the reduced cost of original variables.
- application: binary multicommodity flow problem

Binary multicommodity flow problem: subproblem

$$\min \quad \sum_{k\in K}\sum_{p\in P^k} q^k c_p^k y_p^k \tag{11}$$

$$\text{subj. to} \quad \sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p q^k y_p^k \le u_{ij}\ ,\ \forall (i,j)\in A \tag{12}$$

$$\sum_{p\in P^k} y_p^k = 1,\ \forall k\in K \tag{13}$$

$$y_p^k \in \{0,1\},\ \forall p\in P^k, \forall k\in K \tag{14}$$

- $-\pi_{ij}$: nonnegative dual variables associated to constraints (12)
- σ^k : unrestricted dual variables associated to constraints (13)
- coefficient q^k in all constraints (12), *i.e.*, in all the arcs of the path.

Reduced cost of column p :

$$\begin{aligned}\overline{c}_p^k &= \sum_{(i,j)\in A} q^k \delta_{ij}^p c_{ij}^k - \Big(\sum_{ij\in A} q^k(-\pi_{ij})\delta_{ij}^p + \sigma^k\Big) = \\ &= \sum_{ij\in A} q^k(c_{ij}^k + \pi_{ij})\delta_{ij}^p - \sigma^k,\ \forall p\in P^k, \forall k\in K.\end{aligned}$$

Solution shown in Example with 5 commodities

	y_1^{blue}	y_1^{orange}	y_1^{green}	y_1^{red}	y_1^{yellow}		
(1,2)	10					$\leq$	10
(1,3)		7				$\leq$	12
(2,4)	10				11	$\leq$	32
(2,5)			8			$\leq$	8
(3,5)		7		5		$\leq$	12
(4,6)				5	11	$\leq$	16
(5,6)		7		5	11	$\leq$	24
1	1					$=$	1
2		1				$=$	1
3			1			$=$	1
4				1		$=$	1
5					1	$=$	1
min	20	14	16	10	22		

Binary multicommodity flow problem: subproblem (cont.)

Solve one subproblem for each commodity k :

Shortest path problem with arcs with reduced costs $c_{ij}^k + \pi_{ij}$, for each commodity k.

adjust reduced cost of optimal path p^* using q^k and use dual information σ^k to evaluate column p^* :

Column p^* is attractive for commodity k, if

$$c_{p^*}^k q_k - \sigma^k < 0$$

Knapsack constraints and cover inequalities

Given a knapsack constraint

$$S = \{x \in \{0,1\}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b, \forall i\}.$$

$C \subseteq \{1,\ldots,n\}$ is a *cover* if $\sum_{i \in C} a_i > b$.

Cover inequalities $\sum_{i \in C} x_i \leq |C| - 1$ are valid inequalities for the integer problem.

A cover is *minimal* if, for each $k \in C, (\sum_{i \in C} a_i) - a_k \leq b$.

Example	knapsack constraint						
	$9x_1$	$+7x_2$	$+6x_3$	$+4x_4$	$+4x_5$	$\leq$	12

cover	some minimal cover inequalities						
$\{1,2\}$	x_1	$+x_2$				$\leq$	1
$\{1,3\}$	x_1		$+x_3$			$\leq$	1
$\{2,3\}$		x_2	$+x_3$			$\leq$	1
$\{2,4,5\}$		x_2		$+x_4$	$+x_5$	$\leq$	2

Lifted cover inequalities (LCI)

Given a cover inequality for a cover C (and $\overline{C} = \{1,\ldots,n\} \setminus C$), a stronger inequality can be found:

lifting:

determining the largest coefficients α_i, $\alpha_i \geq 0$, in

$$\sum_{i \in C} x_i + \sum_{i \in \overline{C}} \alpha_i x_i \leq |C| - 1$$

such that the LCI is still valid for set S.

each element in $\overline{C}$, is lifted, one at a time, in a pre-chosen sequence, by solving a series of knapsack problems (one for each member of $\overline{C}$).

Different LCIs can be found depending on the sequence

Finding the most violated LCI is NP-hard [Gu *et al.* '95].

Lifted cover inequalities(cont.)

Example	knapsack constraint						
	$9x_1$	$+7x_2$	$+6x_3$	$+4x_4$	$+4x_5$	$\leq$	12

cover	minimal cover inequality				
$\{2,4,5\}$	x_2	$+x_4$	$+x_5$	$\leq$	2

First element in sequence: lifting α_1

$$\alpha_1 x_1 + x_2 + x_4 + x_5 \leq 2$$

- if $x_1 = 0$, the inequality is valid $\forall \alpha_1$.
- if $x_1 = 1$, the inequality is valid if and only if:

$$\alpha_1 + x_2 + x_4 + x_5 \leq 2$$

- is valid for all $\{x_2, x_4, x_5\} \in \{0,1\}^3$ that obey the constraint

$$7x_2 + 4x_4 + 4x_5 \leq 12 - 9$$

Lifted cover inequalities(cont.)

Example	knapsack constraint						
	$9x_1$	$+7x_2$	$+6x_3$	$+4x_4$	$+4x_5$	$\leq$	12

cover	minimal cover inequality				
$\{2,4,5\}$	x_2	$+x_4$	$+x_5$	$\leq$	2

First element in sequence: lifting α_1 (cont.)

- $\alpha_1 x_1 + x_2 + x_4 + x_5 \leq 2$
- The maximum value that $x_2 + x_4 + x_5$ can take when $7x_2 + 4x_4 + 4x_5 \leq 12 - 9$ is 0.
- So, α_1 can be equal to 2.
- Problem solved is to find largest α_1 :

$$\alpha_1 + max\{x_2 + x_4 + x_5 : 7x_2 + 4x_4 + 4x_5 \leq 12 - 9, x \in \{0,1\}^3\} \leq 2$$

Lifted cover inequalities(cont.)

Example	knapsack constraint						
	$9x_1$	$+7x_2$	$+6x_3$	$+4x_4$	$+4x_5$	$\leq$	12

cover	LCI after lifting α_1					
$\{2,4,5\}$	$2x_1$	$+x_2$		$+x_4$	$+x_5$	$\leq$	2

Second element in sequence: lifting α_3

- $2x_1 + x_2 + \alpha_3 x_3 + x_4 + x_5 \leq 2$
- The maximum value that $2x_1 + x_2 + x_4 + x_5$ can take when $9x_1 + 7x_2 + 4x_4 + 4x_5 \leq 12 - 6$ is 1.
- So, α_3 can be equal to 1.
- Problem solved is to find largest α_3 :

$$\alpha_3 + max\{2x_1 + x_2 + x_4 + x_5 : 9x_1 + 7x_2 + 4x_4 + 4x_5 \leq 12 - 6, x \in \{0,1\}^4\} \leq 2$$

Lifted cover inequalities(cont.)

Example	knapsack constraint						
	$9x_1$	$+7x_2$	$+6x_3$	$+4x_4$	$+4x_5$	$\leq$	12

cover	LCI with sequence 1, 3						
$\{2,4,5\}$	$2x_1$	$+x_2$	$+x_3$	$+x_4$	$+x_5$	$\leq$	2

Application: Binary multicommodity flow problem

Barnhart *et al.* '00:

Constraints in the original arc-flow model are knapsack constraints:

$$\sum_{k \in K} q^k x_{ij}^k \leq u_{ij}, \ \forall (i,j) \in A$$

and LCI can be derived:

$$\sum_{k \in C} x_{ij}^k + \sum_{k \in \overline{C}} \alpha_k x_{ij}^k \leq |C| - 1$$

- The LCI is expressed in terms of the original arcflow variables.
- Each LCI is associated with a given arc (i,j)

Application: Binary multicommodity flow problem

- Given $x_{ij}^k = \sum_{p \in P} y_p^k \delta_{ij}^p$,
- the LCI $\sum_{k \in C} x_{ij}^k + \sum_{k \in \overline{C}} \alpha_k x_{ij}^k \leq |C| - 1$
- can also be expressed in terms of the (reformulated) path model:

$$\sum_{k \in C} 1 \sum_{p \in P} y_p^k \delta_{ij}^p + \sum_{k \in \overline{C}} \alpha_k \sum_{p \in P} y_p^k \delta_{ij}^p \leq |C| - 1$$

- The LCI for a given arc (p, q) involves all commodities.
- It is explicitly added to the restricted master problem.

Application: Binary multicommodity flow problem

Transferral of dual information of

$$\sum_{k \in C} 1 \sum_{p \in P} y_p^k \delta_{ij}^p + \sum_{k \in \overline{C}} \alpha_k \sum_{p \in P} y_p^k \delta_{ij}^p \leq |C| - 1 :$$

- $-\gamma_{pq}$: dual variable associated to a given LCI associated to arc (p, q).
- We can anticipate, while solving the subproblem, which will be the coefficient of the new column in the RMP, for each LCI previously added in the RMP, for each commodity k. It will be α_{pq}^k (where $\alpha_{pq}^k = 1$, if $k \in C$).
- Dual information of the LCI is used to change the reduced cost of *one* arc (p, q) for commodity k

$$\overline{c}_{pq}^k = c_{pq}^k + \pi_{pq} + \alpha_{pq}^k \gamma_{pq}, \ \forall k \in K,$$

- leading to the reduced cost of path.

Binary multicommodity flow problem: remarks

- Use of cuts dramatically decreased the number of nodes searched in the instances for which the optimal solution was found.
- For most of the other instances, it decreased the optimality gap.

Other types of cuts

- Supperadditive functions
- Dual feasible functions (DFF)
- Primal cuts derived from dual feasible functions (maximal DFF)
- Their use in column generation
- Application in the minimization of number of set-ups of the CSP

Superadditive functions [Nemhauser,Wolsey,1988]

- A function $F : D \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^1$ is called *superadditive* if

$$F(d_1) + F(d_2) \leq F(d_1 + d_2), \quad \forall d_1, d_2, d_1 + d_2 \in D.$$

- A function $F : D \rightarrow \mathbb{R}^1$ is called *nondecreasing* over D if $d_1, d_2 \in D$ and $d_1 \leq d_2$ implies $F(d_1) \leq F(d_2)$.
- Assume $F(0) = 0$ and $0 \in D$

Nice result

Nemhauser,Wolsey'88: Every valid inequality for a nonempty set $S = \mathbb{Z} \cap \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ is equivalent to or dominated by a superadditive valid inequality.

Superadditive valid inequalities [Vanderbeck'2000]

- Superadditive nondecreasing function $F^\gamma(z) = \max\{0, \lceil \frac{\gamma z}{b} \rceil - 1\}$, with $\gamma \in \{2, \ldots, b\}$
- Superadditive inequalities

$$\sum_i \left(\left\lceil \frac{\gamma a_i}{b} \right\rceil - 1 \right) x_i \leq \gamma - 1,$$

are valid inequalities for

$$X = \{x \in \mathbb{N}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b, \forall i\}.$$

Example

$$9x_1 + 7x_2 + 6x_3 + 4x_4 + 2x_5 \leq 12$$

a_i/b	0.75	0.58	0.5	0.33	0.17		
$\gamma = 2$	$1x_1$	$+1x_2$				$\leq$	1
$\gamma = 3$	$2x_1$	$+1x_2$	$+1x_3$			$\leq$	2
$\gamma = 4$	$2x_1$	$+2x_2$	$+1x_3$	$+1x_4$		$\leq$	3

Dual feasible functions (DFF)

A function $f : [0,1] \rightarrow [0,1]$ is said to be dual feasible if, for any finite set of real numbers $S \subseteq [0,1]$,

$$\sum_{x \in S} x \leq 1 \Rightarrow \sum_{x \in S} f(x) \leq 1$$

Applications

Lower bounds in combinatorial enumeration algorithms:

Fekete,Schepers'97: Let $I := (x_1, \ldots, x_n)$ be a BPP instance and let u be a DFF. Then, any lower bound for the transformed BPP instance $u(I) := (u(x_1), \ldots, u(x_n))$ is also a lower bound for I.

Redundant functions (Carlier,Néron'2005)

Independently, a discrete version was also proposed:
A function $f : [0, W] \rightarrow [0, W']$, W and W' integers, is said to be a redundant function, or discrete dual feasible function, if, for any

$$w_1 + w_2 + \ldots + w_k \leq W \Rightarrow f(w_1) + f(w_2) + \ldots + f(w_k) \leq f(W) = W'$$

Applications

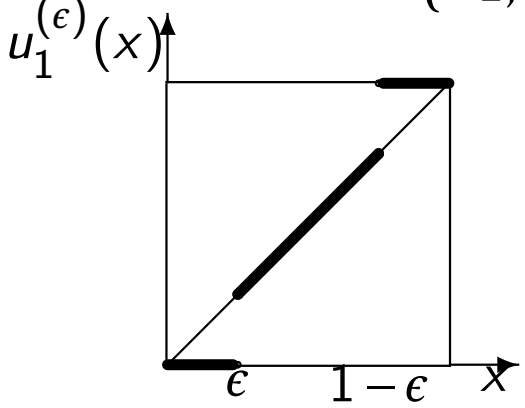
Cuts for integer programming in LP based optimization:

Alves,VC'2008: if integer solution obeys initial constraint, it will also obey modified constraint; possibly, fractional solutions are cut.

Dual feasible functions: example 1 [Fekete,Schepers,2001]

Let $\epsilon \in \left[0, \frac{1}{2}\right]$ $\qquad u_1^{(\epsilon)}: \quad [0,1] \to [0,1]$

$$x \mapsto \begin{cases} 0, & \text{for } x < \epsilon, \\ x, & \text{for } \epsilon \le x \le 1-\epsilon, \\ 1, & \text{for } x > 1-\epsilon, \end{cases}$$



Formalizes Martello and Toth's lower bound L2 for bin-packing.

Proposition (Alves,VC'2008)

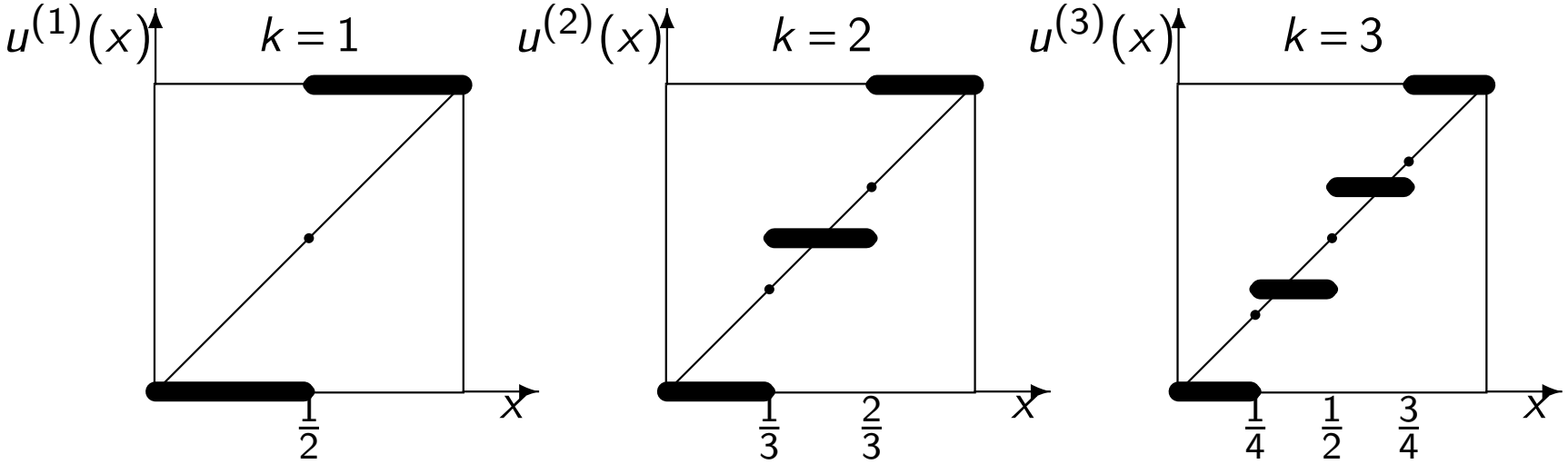
Function $u_1^{(\epsilon)}$ is nondecreasing and superadditive over $[0,1]$, for $\epsilon \in [0, \frac{1}{2}]$.

Dual feasible functions: example 2 [Fekete,Schepers'2001]

DFF $u^{(k)}$, $k \in \mathbb{N}$, slightly improves a function proposed earlier by Lueker'83.

$$u^{(k)}: \quad [0,1] \to [0,1]$$

$$x \mapsto \begin{cases} x, & \text{for } (k+1)x \in \mathbb{Z}, \\ \frac{\lfloor (k+1)x \rfloor}{k}, & \text{otherwise.} \end{cases}$$



Superadditive valid inequalities from DFF

Function $u^{(k)}$ is nondecreasing.

Proposition (Alves,VC'2008)

For $k \in \mathbb{N}$, $u^{(k)}$ is superadditive over $[0,1]$.

Proof: Omitted $\square$

Proposition

The inequality

$$\sum_i u^{(k)}\left(\frac{a_i}{b}\right) x_i \leq 1, \quad k \in \mathbb{N}$$

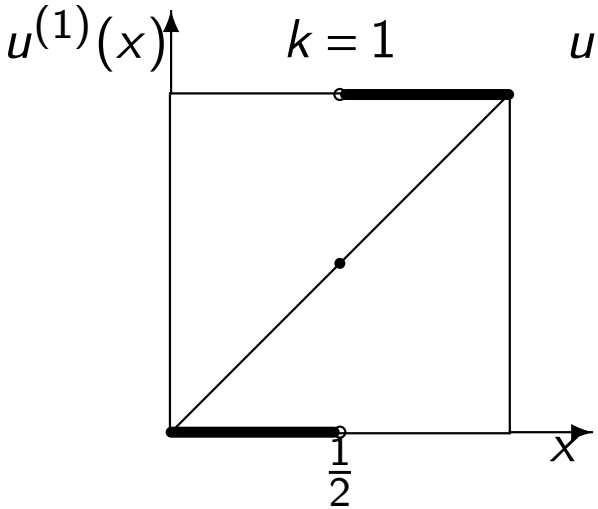
is a valid inequality for integer knapsack polytopes
$S = \{x \in \mathbb{N}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b,\ \forall i\}$.

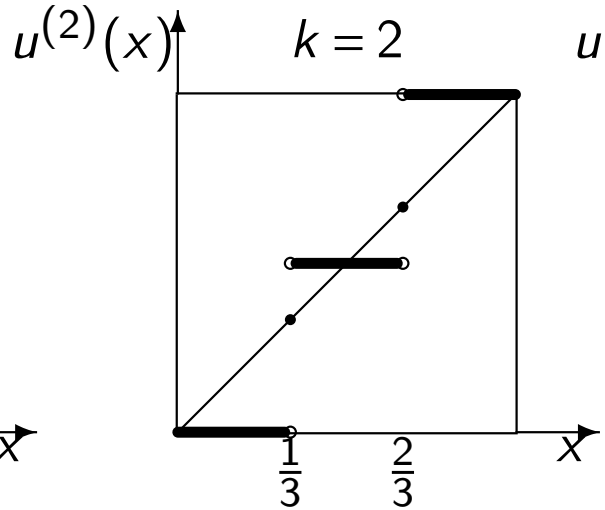
Proof: It is superadditive and nondecreasing (follows from Nemhauser,Wolsey'88). $\square$

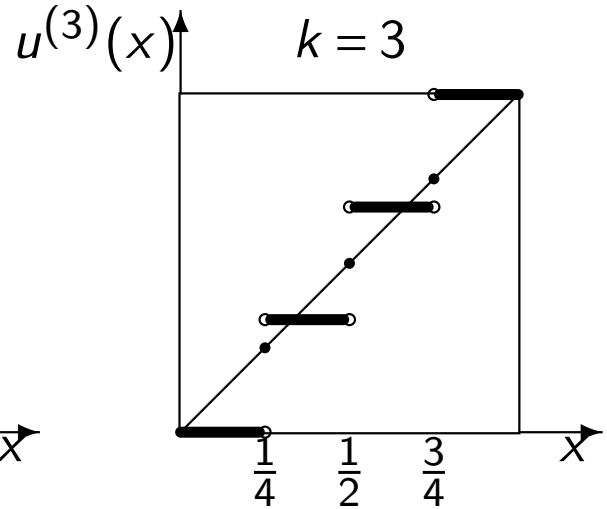
Valid inequalities from DFF: example

$$9x_1 \quad +7x_2 \quad +6x_3 \quad +4x_4 \quad +2x_5 \quad \leq \quad 12$$

a_i/b	0.75	0.58	0.5	0.33	0.17		
$k=1$	$1x_1$	$+1x_2$	$+\frac{1}{2}x_3$			$\leq$	1
$k=2$	$1x_1$	$+\frac{1}{2}x_2$	$+\frac{1}{2}x_3$	$+\frac{1}{3}x_4$		$\leq$	1
$k=3$	$\frac{3}{4}x_1$	$+\frac{2}{3}x_2$	$+\frac{1}{2}x_3$	$+\frac{1}{3}x_4$		$\leq$	1







Proposition

The inequality

$$\sum_i u^{(k)}\left(\frac{a_i}{b}\right) x_i \leq 1, \quad k \in \mathbb{N}$$

is a valid inequality for integer knapsack polytopes that is equivalent or dominates

$$\sum_i \left(\left\lceil \frac{\gamma a_i}{b} \right\rceil - 1\right) x_i \leq \gamma - 1, \quad \gamma \in \{2, \ldots, b\}.$$

Proof: Let $z_i = \frac{a_i}{b}$. Vanderbeck's inequalities can be rewritten as:
$\sum_i \frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1} x_i \leq 1$.
Let $k = \gamma - 1$.
For $\gamma z_i \notin \mathbb{Z}$, we have $u^{(k)}(z_i) = \frac{\lfloor (k+1) z_i \rfloor}{k} = \frac{\lfloor \gamma z_i \rfloor}{\gamma - 1} = \frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1}$.
For $\gamma z_i \in \mathbb{Z}$, we have
$u^{(k)}(z_i) = z_i \geq \frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1} = \frac{\lceil (k+1) z_i \rceil - 1}{k} = \frac{(k+1) z_i - 1}{k} = z_i + \frac{z_i}{k} - \frac{1}{k}$, since $z_i \leq 1$.
□

Comparative strengths: example (cont.)

	$9x_1$	$+7x_2$	$+6x_3$	$+4x_4$	$+2x_5$	$\leq$	12
$\gamma = 2$	$1x_1$	$+1x_2$				$\leq$	1
$\gamma = 3$	$1x_1$	$+\frac{1}{2}x_2$	$+\frac{1}{2}x_3$			$\leq$	1
$\gamma = 4$	$\frac{2}{3}x_1$	$+\frac{2}{3}x_2$	$+\frac{1}{3}x_3$	$+\frac{1}{3}x_4$		$\leq$	1
$k = 1$	$1x_1$	$+1x_2$	$+\frac{1}{2}x_3$			$\leq$	1
$k = 2$	$1x_1$	$+\frac{1}{2}x_2$	$+\frac{1}{2}x_3$	$+\frac{1}{3}x_4$		$\leq$	1
$k = 3$	$\frac{3}{4}x_1$	$+\frac{2}{3}x_2$	$+\frac{1}{2}x_3$	$+\frac{1}{3}x_4$		$\leq$	1

Maximal DFF (MDFF) [Carlier, Néron'02]

MDFF are not dominated by any other valid DFF.

Definition

Let f and f' be two DFF. f is dominated by f' ($f \leq f'$) if $\frac{f(c_i)}{f(C)} \leq \frac{f'(c_i)}{f'(C)}$ for all $c_i \leq C$. Moreover if there exists a value j such that $\frac{f(c_j)}{f(C)} < \frac{f'(c_j)}{f'(C)}$, f is strictly dominated by f' ($f < f'$).

Definition

A DFF f is an MDFF if there does not exist any DFF f' such that $f' > f$.

Characterization of Maximal DFF

Proposition (Carlier, Nerón '02)

A DFF f is an MDFF if and only if the following conditions hold:

- *f is nondecreasing,*
- *f is superadditive,*
- *f is symmetric* ($\forall x \in [0, C], f(x) + f(C - x) = f(C)$),
- $f(0) = 0$.

A way of creating a MDFF

Theorem

Let f be a superadditive and nondecreasing function defined from $[0, C]$ to $[0, f(C)]$, and such that $f(0) = 0$. The following function is a maximal DFF.

$$\widehat{f} : [0, C] \rightarrow [0, 2f(C)]$$

$$x \mapsto \begin{cases} 2f(C) - 2f(C-x), & \text{for } C \geq x > \frac{C}{2}, \\ f(C), & \text{for } x = \frac{C}{2}, \\ 2f(x), & \text{for } x < \frac{C}{2}. \end{cases}$$

Proof: *omitted.*

Using this result, many well-known superadditive functions can lead to MDFF.

Note that $\widehat{f}$ dominates f, but it is not the only function to dominate f.

Compositions and linear combinations of DFF and MDFF

Proposition (Nemhauser, Wolsey '88)

A composition, or a positive linear combination of superadditive functions is superadditive.

If f and g are superadditive, then λf, $\lfloor f \rfloor$, $f+g$, $min\{f,g\}$ are superadditive.
Some results concerning the *safe* operations for superadditivity also hold for MDFF (sum, composition), while other do not (min, for example).
These properties hold when discrete functions are considered.

Proposition

If f and g are MDFF, then $f+g$, λf and $f \circ g$ are MDFF.

Branch-and-price-and-cut methodology

- Given solution of restricted master problem at a given node of the search tree:
- Using one, or several, families of cuts (*e.g.*, depending on k).
- Find most (or, first, or other) violated cut
- Add to model, and re-optimize
- Proceed with column generation

Using dual feasible functions to derive cuts

- In the subproblem, we are able to anticipate the coefficient that the column will have in the primal cut,
- because it comes from mapping the value of the coefficient of a constraint of the original model, and
- so we can correctly evaluate the attractiveness of the column.

In the setting of column generation, for general problems, with no special structure, cuts from DFF may be useful in strengthening models.

Minimization of number of set-ups: model

First, solve CSP to find z^*_{CSP}.
z^*_{CSP} : minimum number of rolls needed.
Then, the model is:

$$
\begin{aligned}
\min\ z_{IP} \quad &= \quad \sum_{j \in J} \delta(x_j) \\
\text{subj. to} \quad & \sum_{j \in J} a_{ij} x_j = b_i,\ \ i = 1, 2, \ldots, m \\
& \sum_{j \in J} x_j \leq z^*_{CSP} \\
& x_j \geq 0 \text{ and integer },\ \ \forall j \in J
\end{aligned}
$$

$$
\delta(x_j) = \begin{cases} 1, & \text{if } x_j > 0 \\ 0, & \text{if } x_j = 0 \end{cases}
$$

Model for min. number of set-ups [Vanderbeck,2000]

λ_{jn} : usage of the pattern j replicated n times (pattern multiplicity)
$u(J_j)$: maximum value of n for a pattern j, equal to $\min_{i=1,\ldots,m} \left\lfloor \frac{b_i}{a_{ij}} \right\rfloor$.

$$
\begin{aligned}
\min \quad & \sum_{j \in J} \sum_{n=1}^{u(J_j)} \lambda_{jn} \\
\text{subj. to} \quad & \sum_{p \in P} \sum_{n=1}^{u(J_j)} n a_{ip} \lambda_{jn} = b_i, \ i = 1, \ldots, m, \\
& \sum_{j \in J} \sum_{n=1}^{u(J_j)} n \lambda_{jn} \leq z^*_{CSP}, \\
& \lambda_{jn} \in \{0,1\}, \ j \in J, \ n = 1, \ldots, u(J_j).
\end{aligned}
$$

Cuts used for integer knapsack polytope.

Strengthening Vanderbeck's model

For cutting pattern j, the associated trim loss T_j is:

$$T_j = W - \sum_{i=1}^{m} a_{ij} w_i.$$

New waste constraint added to the model:

$$\sum_{p \in P} \sum_{n=1}^{u(J_j)} n T_j \lambda_{jn} = z_{CSP} W - \sum_{i=1}^{m} w_i b_i$$

At optimum, all constraints are obeyed without slack.
Cuts from DFF derived from:

- demand constraints
- maximum number of rolls constraint
- waste constraint

Minimization of number of set-ups: example

Example similar to one used before, but $b_3 = 7$.
Integer solution of CSP $z^*_{CSP} = 6 \Rightarrow$ loss$= 2$.

	λ_{11}	λ_{12}	λ_{21}	λ_{22}	λ_{31}	λ_{32}	λ_{33}	λ_{41}	λ_{42}	λ_{51}	λ_{52}	λ_{61}	
$w_i = 4$	2	4	1	2	1	2	3						$= 5$
3			1	2				2	4	1	2		$= 4$
2					2	4	6	1	2	2	4	4	$= 7$
loss $= 1$			1	2						1	2		$= 2$
z^*_{CSP}	1	2	1	2	1	2	3	1	2	1	2	1	≤ 6
min	1	1	1	1	1	1	1	1	1	1	1	1	

Non-maximal (with loss=2) patterns of multiplicity 1 are not represented.
Cut derived from demand constraint $w_3 = 2$ with DFF $u^{(1)}(x)(k = 1)$:

	λ_{11}	λ_{12}	λ_{21}	λ_{22}	λ_{31}	λ_{32}	λ_{33}	λ_{41}	λ_{42}	λ_{51}	λ_{52}	λ_{61}	
$k = 1$						1	1				1	1	≤ 1

Primal cuts: some computational results

Comparison for 16 instances from real-life problems [Vanderbeck,2000]

Time limit: 2 hours.
Values of the LP optima are improved by 21.5%, at root node.
60% less nodes in average than [Vanderbeck, 2000].
Still, only one more instance is solved (13, instead of 12).

3600 instances [Alves, VC,2008]

m	average demand	instances solved	average optimality gap
20	10	100 %	
20	20 or 30	89 %	$\leq$ 3%
30		57.6 %	8.9 %
40		33.5 %	12.1 %

Time limit: 10 minutes.

Minimization of number of set-ups: remarks

- Problem of minimizing the number of set-ups in the CSP is very hard.
- Looking at ties between polyhedral theory and DFF may help in getting more useful cuts.

Cuts expressed in terms of the reformulated variables

- Some valid inequalities are powerful for set partitioning problems

may not be so easy to deal with if they are expressed in terms of the reformulated variables

- example: constraints for set partitioning (reformulated) problem,
- structure of subproblem is modified.
- reference: vehicle routing problem

Set packing problems

- S : finite set with m elements,
- $S_1, S_2, \ldots, S_n$, a collection of subsets of S.
- A packing of S is a collection of subsets, $S_{i_1}, \ldots, S_{i_j}, \ldots, S_{i_K}$, identified by $i_1, \ldots, i_j, \ldots, i_K$, such that:

$$\begin{aligned} \cup_{j=1}^{k} S_{i_j} &\subseteq S \\ S_{i_i} \cap S_{i_j} &= \emptyset, \ \forall i, j \end{aligned}$$

- The set of solutions of a packing problem is a relaxation of the set of solutions of the corresponding partitioning problem.
- Every valid inequality for a packing problem is also valid for the corresponding partitioning problem.

Valid inequalities

Graph $G = (V, A)$ where arc $(i, j) \in A$ if i and j are conflicting vertices.

clique inequalities:

- clique K_p is a complete subgraph with p vertices.

$$\sum_{i \in K_p} x_i \leq 1$$

odd hole inequalities:

- cycle C_p with an odd number, p, of vertices.

$$\sum_{i \in C_p} x_i \leq \lfloor p/2 \rfloor$$

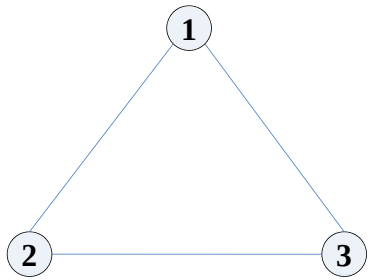
Valid inequalities for set packing problems

Set packing problem

$$\max\{cx : Ax \leq 1, x \in \{0,1\}^n\}, \text{ and } A \in \{0,1\}^{m \times n}.$$

Example:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	1	1		1			1	≤ 1
2		1	1		1		1	≤ 1
3	1		1			1	1	≤ 1
solution:	0.5	0.5	0.5					



cycle C_3 inequality:

- $x_1 + x_2 + x_3 \leq 1$

Also happens to be a clique inequality.

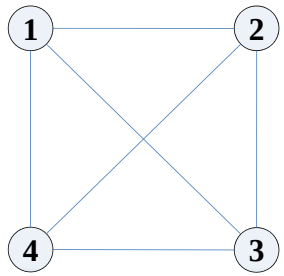
Valid inequalities for set packing problems (cont.)

Set packing problem

$$\max\{cx : Ax \leq 1, x \in \{0,1\}^n\}, \text{ and } A \in \{0,1\}^{m \times n}.$$

Example:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	1	1	1				1	≤ 1
2	1			1	1			≤ 1
3		1		1		1	1	≤ 1
4			1		1	1	1	≤ 1
solution:	0.33	0.33	0.33	0.33	0.33	0.33		



clique K_4 inequality:

- $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 1$

Reference: vehicle routing problem with time windows

- Exploits structures of the reformulated problem
- Handled successfully, because valid inequalities are expressed in terms of combinations of other rows.
- Subset row inequalities (SR) are Chvátal-Gomory rank-1 cuts.
- pricing problem becomes a shortest-path problem with nonadditive nondecreasing constraints or objective function
- can be handled modifying the dominance criteria, in the label-setting algorithm.
- open instances solved to optimality.

M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger, "Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows", Operations Research, Vol. 56, No. 2, pp. 497Ű511, 2008.

Concluding remark

- Primal cuts are as important in branch-and-price as they are in branch-and-cut.

Part VIII

Lower Bounds and Heuristics

Outline

- Lower bounds
  - Trivial lower bound
  - Farley's bound
  - Lagrangean bound
  - Lasdon bound
  - bounds from Dual Feasible Functions (in Part VII)

Lower bounds for CSP

Duality is a tool for deriving lower bounds (for minimization problems):

- Find a feasible dual solution, which obeys all dual constraints corresponding to all valid columns.
- Using the weak duality theorem:
- The objective function value of the feasible dual solution is a Lower Bound to the value of the optimal solution of the minimization problem.

Examples:

- Trivial lower bound
- Farley's bound
- Lagrangean bound
- bounds from Dual Feasible Functions (in Part VI)

Trivial lower bound

Equivalent to Martello and Toth's $LB1$.

Dual constraints are:

$$\sum_{i=1}^{m} a_{ij}\pi_i \leq 1, \forall j \in J.$$

Dual solution $\overline{\pi}_i = w_i/W, \forall i$, is a feasible dual solution, because

$$\sum_{i=1}^{m} a_{ij} w_i \leq W$$
$$a_{ij} \geq 0 \text{ and integer, } \forall j \in J.$$

The objective function value of this feasible dual solution is

$$\sum_{i=1}^{m} b_i w_i/W.$$

Farley's lower bound (Farley'1992) - I

At any iteration: the current dual solution, $\overline{\pi}$, divided by the value of the optimum of subproblem is a dual feasible solution.

"Most attractive" column A^{min} has minimum reduced cost:

$$\overline{c}_{min} = 1 - \overline{\pi} A^{min} = \min_{j \in J}(1 - \overline{\pi} A_j).$$

$$1 - \overline{\pi} A^{min} \leq 1 - \overline{\pi} A_j, \forall j \in J,$$

$$\overline{\pi} A_j \leq \overline{\pi} A^{min}, \forall j \in J,$$

$$(\overline{\pi} / \overline{\pi} A^{min}) A_j \leq 1, \forall j \in J.$$

Farley's lower bound (Farley'1992) - II

So, $(\overline{\pi}/\overline{\pi}A^{min})$ is a feasible solution to the dual of the CSP.

The objective function value of this feasible dual solution, $(\overline{\pi}/\overline{\pi}A^{min})b$, is a lower bound.

For practical purposes, at each iteration of the column generation process, calculate it as follows:

- divide $\overline{\pi}b$ by $\overline{\pi}A^{min}$.
-
- $\overline{\pi}b$: value of the optimal current primal solution.
- $\overline{\pi}A^{min}$: the optimal value of the knapsack subproblem.
-
- Easy to calculate, can be updated at each iteration.

Lagrangean lower bound

Lagrangean function of the LP relaxation of CSP is:

$$\sum_{j \in J} x_j + \sum_{i=1}^{m} \overline{\pi}_i (b_i - \sum_{j \in J} a_{ij} x_j)$$

$$\sum_{i=1}^{m} \overline{\pi}_i b_i + \sum_{j \in J} (1 - \sum_{i=1}^{m} \overline{\pi}_i a_{ij}) x_j$$

$$\overline{\pi} b + \sum_{j \in J} \overline{c}_j x_j .$$

$\overline{c}_j = 1 - \sum_{i=1}^{m} \overline{\pi}_i a_{ij}$: reduced cost of variable x_j

Let K be an upper bound on $\sum_{j \in J} x_j$.

Lagrangean lower bound (cont.)

As $x_j \geq 0$ and $\overline{c}_j \geq \overline{c}_{min}, \forall j \in J$:

$$\overline{\pi} b + \sum_{j \in J} \overline{c}_j x_j \geq \overline{\pi} b + \overline{c}_{min} \sum_{j \in J} x_j.$$

As $0 \leq \sum_{j \in J} x_j \leq K$ and $\overline{c}_{min} \leq 0$:

$$\overline{\pi} b + \sum_{j \in J} \overline{c}_j x_j \geq \overline{\pi} b + \overline{c}_{min} K.$$

From lagrangean duality,

$$z_{LP} \geq \min_{x \geq 0} \overline{\pi} b + \sum_{j \in J} \overline{c}_j x_j.$$

Hence,

$$\overline{\pi} b + \overline{c}_{min} K \leq z_{LP}.$$

Leftmost term of this relation is called the lagrangean bound.
Substituting the optimal value of the linear relaxation of CSP, z_{LP}, for K (actually, z_{LP} is *equal* to the optimal value of $\sum_{j \in J} x_j$), we obtain a bound equal to Farley's bound: $\overline{\pi} b / (1 - \overline{c}_{min}) \leq z_{LP}$.

Lasdon bound

Problem with $|K|$ convexity constraints.

$$\min z = \sum_{i,j} c_{ij} x_{ij} \tag{15}$$

$$\text{subj. to} \quad \sum_{i,j} a_{ij} x_{ij} = b_0 \tag{16}$$

$$\sum_{j} x_{ij} = 1, \ i = 1,2,\ldots,K \tag{17}$$

$$x_j \geq 0, \ \forall i,j \tag{18}$$

π_0 : dual variable of constraint (16).
$\pi_i, \ i = 1,2,\ldots,K$: dual variables of constraints (17).
(15) - π_0 (16) - π_i (17)

$$z - \pi_0 b_0 - \sum_i \pi_i = \sum_{i,j} x_{ij}(c_{ij} - \pi_0 a_{ij} - \pi_i)$$

Lasdon bound (cont.)

$\overline{c}_{ij} = c_{ij} - \pi_0 a_{ij} - \pi_i$: reduced cost of variable x_{ij}, which is attractive when $\overline{c}_{ij} < 0$ (minimization problem).
Solving the $|K|$ subproblems, we obtain $\min \overline{c}_{ij}$ for each i.

$$z - \pi_0 b_0 - \sum_i \pi_i \geq \sum_{i,j} x_{ij} (\min_j \overline{c}_{ij})$$

$$z - \pi_0 b_0 - \sum_i \pi_i \geq \sum_i (\min_j \overline{c}_{ij}) \sum_j x_{ij}$$

$$z \geq \pi_0 b_0 + \sum_i \pi_i + \sum_i (\min_j \overline{c}_{ij})$$

Sum of first two terms on the right side is the value of dual solution of the current restricted master problem z_B.

$$z \geq z_B + \sum_i (\min_j \overline{c}_{ij})$$

At the optimum solution, $\overline{c}_{ij} \geq 0, \ \forall i$
Bound can be computed with little computational effort.

Outline

- Heuristics
  - Quality of models and quality of heuristics
  - Rounding heuristics: cutting stock problem
  - Local search based on column generation
  - Application: binary multicommodity flows

Quality of models and quality of heuristics

Heuristics based on stronger models generally provide better results than heuristics based on weak models.

General drawbacks of heuristics and meta-heuristics

- lack of a measure of the quality of the solution obtained
- lack of a well defined stopping criterion.

Rounding LP solution heuristic

- Round up fractional values of LP relaxation optimum to obtain integer solution.
- Value of heuristic solution $z_H : z_H \leq z_{LP} + m$, where z_{LP} is the optimum value of the LP relaxation.
- As the values of the demands are generally high, the integer solution is of good quality.

[Gilmore,Gomory'61]

Rounding heuristic (example)

$W = 8$	cutting patterns						Demand b_i	
	x_1	x_2	x_3	x_4	x_5	x_6		
$w_i = 4$	2	1	1				$\geq$	5
3		1		2	1		$\geq$	4
2			2	1	2	4	$\geq$	8
min	1	1	1	1	1	1		

Optimal fractional solution

2.5		2.0	1.5

6 rolls

Fractional solution rounded up

3.0		2.0	2.0

7 rolls

Excess production: + 1 item of 4 and 2 items of 2.

Heuristics based on column generation

Waescher, Gau'96:

- Pick solution of Gilmore-Gomory model.
- Fix fractional optimal values (several rules to round up or down), thus satisfying an integer part of the demand.
- (remaining unsatisfied demand is a CSP instance with integer demands, called the residual problem).
- Use exact bin packing algorithm to solve residual problem when only few items are left unsatisfied;
- otherwise, use heuristics (FFD and BFD).
- Build solution for the original problem combining the patterns that were fixed with the solution of the residual problem.

Extensive computational experiments with instances with average demands of 10 and 50: optimal solutions were found in almost all cases.

Heuristics based on column generation (cont.)

- Bin packing problem: small demanded quantities: demands close to one or even equal to one.
- Optimal LP relaxation variables are often a fraction of unity, and residual problem may be almost as large as original problem.
- It may not be so easy for heuristics to find good solutions for the residual problems.

Other heuristics:
BISON: Scholl,Klein,Juergens'97.
Alvim,Glover,Ribeiro,Aloise'04

General heuristic based in column generation

- Pick solution of linear programming relaxation.
- Use the column generated at the root node,
- to get an integer solution with an LP software package.

Beam Search

- when exhaustive search in branch-and-bound tree is too heavy,
- explore only a (very) limited number of children of each node.
- Beam width: number of nodes allowed at any level of the breadth-first search tree.
- use breadth first search: all the nodes are evaluated at each level before going any deeper in the search tree.
- use an evaluation function to kill a node or to decide how many children it will generate.

P. Ow, T. Morton, "Filtered beam search in scheduling", International Journal of Production Research, 26, 35-62, 1988.

Beam search-and-price: combine beam search and price at each node for attractive columns.

Combining Meta-heuristics and column generation

- hybrid approach applied to vehicle routing, where the master problem is a set partitioning problem.
- meta-heuristics exclusively based on local search.
- applied in the nodes of the branch-and-price tree for improving the incumbent solution and generating new columns.

E. Danna and C. L. Pape, "Branch-and-Price Heuristics: A Case Study on the Vehicle Routing Problem with Time Windows". In G. Desaulniers, J. Descrosiers, M.M. Solomon (eds.) Column Generation. Springer Science and Business Media, New York, Ch. 4., 2005.

Combining Meta-heuristics and exact methods

Other references

- I. Dumitrescu and T. Stützle, "A survey of methods that combine local search and exact algorithms". In G. R. Raidl, S. Cagnoni, J. J. R. Cardalda, D. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, E. Marchiori, J.-A. Meyer, M. Middendorf (Eds.). "Applications of Evolutionary Computation", Lecture Notes in Computer Science 2611, Springer, pages 211-223, 2003.
- Puchinger, J. and G. R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification". In J. Mira and J. R. Álvarez (Eds.), Lecture Notes in Computer Science 3562, Springer, pages 41-53, 2005.

Concluding remark

- in industrial applications, when there are time limits, heuristics based on column generation may be very competitive.

Bibliography - I

- R.K. Ahuja, T. Magnanti and J.B. Orlin, "Network flows: Theory, Algorithms and Applications". Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.
- S. Martello and P. Toth "Knapsack Problems: Algorithms and Computer Implementations", Wiley, New York, 1990.
- G.L. Nemhauser and L.A. Wolsey, "Integer and Combinatorial Optimization", Wiley Interscience, 1988.
- Column Generation, G. Desaulniers, J. Desrosiers and M.M. Solomon (eds.), Springer, 2005.

Bibliography - II

- C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, "Branch-and-Price: Column Generation for Solving Huge Integer Programs", Operations Research 46(3), 316-329, 1998.
- H. Ben Amor, J. Desrosiers and A. Frangioni "On the Choice of Explicit Stabilizing Terms in Column Generation", Discrete Applied Mathematics, to appear, 2009.
- H. Ben Amor and J.M. Valério de Carvalho, "Cutting Stock Problems", in "Column Generation", G. Desaulniers, J. Desrosiers and M.M. Solomon (eds.), 131-162, Springer, 2005.
- O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck, "Comparison of bundle and classical column generation", Mathematical Programming 113(2), 299-344, 2008.
- S. Fekete and J. Schepers, "New classes of fast lower bounds for bin packing problems". Mathematical Programming 91, 11-31, 2001.
- A. Frangioni "About Lagrangian Methods in Integer Optimization" Annals of Operations Research 139, 163-193, 2005.
- A. Frangioni "Generalized Bundle Methods" SIAM Journal on Optimization 13(1), 117-156, 2002.

Bibliography - III

- O. du Merle, D. Villeneuve, J. Desrosiers and P. Hansen, "Stabilized column generation", Discrete Mathematics 194, 229-297, 1999.
- F. Vanderbeck, "Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem". Operations Research 48(6), 915-926, 2000.
- F. Clautiaux, C. Alves and J.M. Valério de Carvalho, "A survey of dual-feasible and superadditive functions", to appear in Annals of Operations Research, 2009.
- C. Alves and J.M. Valério de Carvalho, "A branch-and-price-and-cut algorithm for the pattern minimization problem", RAIRO Operations Research, 42, 435-453, 2008.
- C. Alves and J.M. Valério de Carvalho, "Stabilized Branch-and-Price -and-Cut Algorithm for the Multiple Length Cutting Stock Problem", Computers and Operations Research 35(4), 1315-1328, 2008.
- F. Alvelos and J.M. Valério de Carvalho, "An extended model and a column generation algorithm for the planar multicommodity flow problem", Networks 50(1), 3-16, 2007.
- M. Lopes and J.M. Valério de Carvalho, "A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times", European Journal of Operational Research 176(3), 1508-1527, 2007.

Bibliography - IV

- C. Alves and J.M. Valério de Carvalho, "Accelerating Column Generation for Variable Sized Bin-Packing Problems", European Journal of Operational Research 183(3), 1333-1352, 2007.
- H. Ben Amor, J. Desrosiers and J.M. Valério de Carvalho, "Dual-optimal Inequalities for Stabilized Column Generation", Operations Research 54(3), 454–463, 2006.
- H. Ben Amor and J.M. Valério de Carvalho, "Cutting Stock Problems", in Column Generation, Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon (eds.), Springer, 2005.
- J.M. Valério de Carvalho, "Using extra dual cuts to accelerate convergence in column generation", INFORMS Journal on Computing 17(2), 175-182, 2005.
- J.M. Valério de Carvalho, "LP Models for Bin-Packing and Cutting Stock Problems", European Journal of Operational Research 141(2), 253-273, 2002.
- 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.