Column Generation
Theory and ~~applications~~ practice

Matteo Salani - matteo.salani@idsia.ch

IDSIA - Dalle Molle Institute for Artificial Intelligence
USI/SUPSI

Workshop on Large Scale Optimization, 2012
Vevey, Switzerland

Context

Large Scale Optimization: Size of input data, Dimension of solution space, Structural properties?

(Non exhaustive) classification of optimization problems:

- Linear/Non-Linear objectives and constraints.
- Continuous/Discrete valued variables.
- Single/Multiple objectives.
- Deterministic/Non Deterministic data

Purpose, desiderata: Feasible/Exact (proof of quality) solutions.

Context

Large Scale Optimization: Size of input data, Dimension of solution space, Structural properties?

(Non exhaustive) classification of optimization problems:

- Linear/Non-Linear objectives and constraints.
- Continuous/Discrete valued variables.
- Single/Multiple objectives.
- Deterministic/Non Deterministic data

Purpose, desiderata: Feasible/Exact (proof of quality) solutions.

Context

Large Scale Optimization: Size of input data, Dimension of solution space, Structural properties?

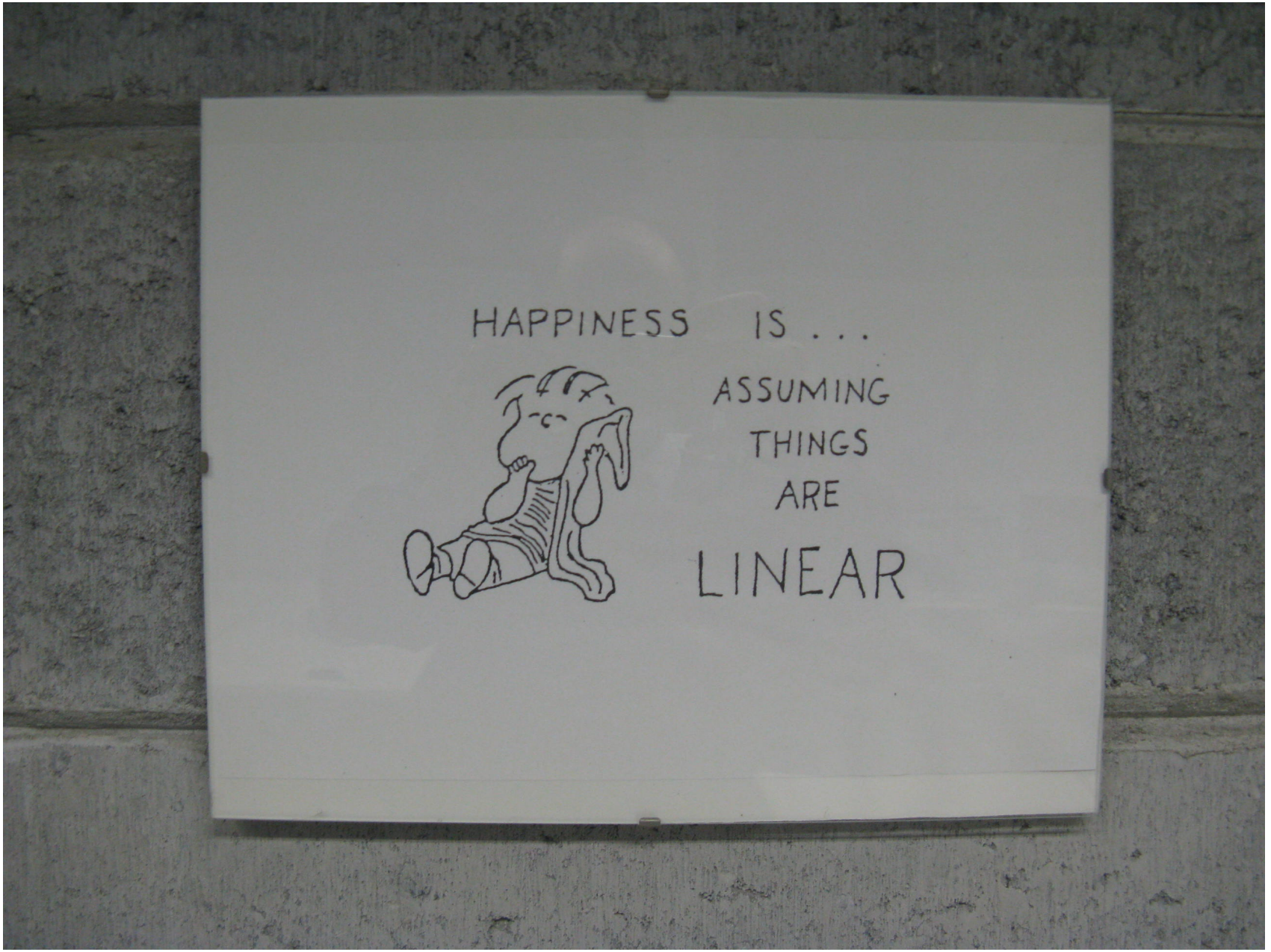
(Non exhaustive) classification of optimization problems:

- Linear/Non-Linear objectives and constraints.
- Continuous/Discrete valued variables.
- Single/Multiple objectives.
- Deterministic/Non Deterministic data

Purpose, desiderata: Feasible/Exact (proof of quality) solutions.

It seems restrictive but a large amount of decision problems can be adequately modeled as such.

Lab's life...



Outline

- Introduction
- Dantzig-Wolfe decomposition
- Illustration
- Integrality and Branch&Price
- Valid inequalities
- Practical implementations with detours in frameworks, prototypes and other funny stories
- Lagrangean and Dual view of CG
- Stabilization and practices
- Advanced topics in CG

Column Generation

An informal definition: an algorithm to deal with many variables that cannot fit in the memory, i.e., cannot be explicitly enumerated.

A suggested computation for maximal multi-commodity network flows. L. R. Ford, D. R. Fulkerson, 1958

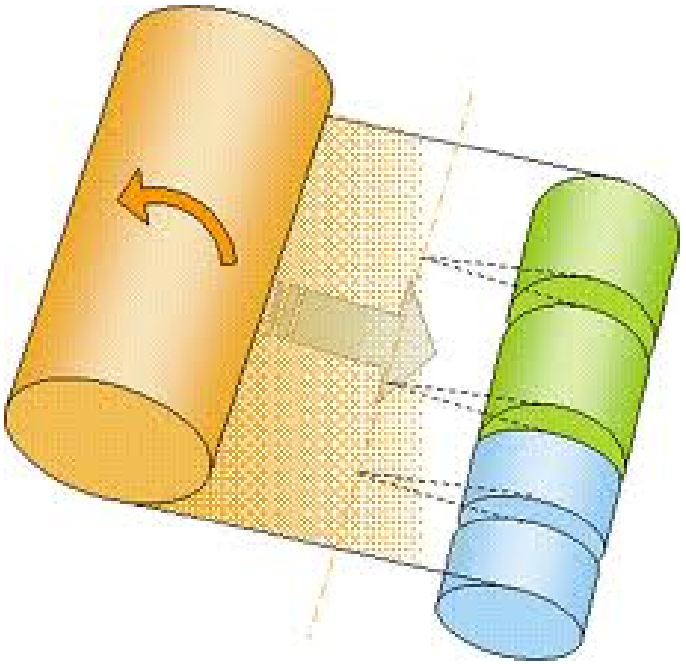
A computation which uses the structure of one formulation of the multi-commodity problem within the framework of a simplex computation to determine maximal multi-commodity flows in networks. For this particular formulation, the number of variables is too large to be dealt with explicitly. The suggested computation treats nonbasic variables implicitly by **replacing the pricing operation of the simplex method with several applications of a combinational algorithm** *for finding a shortest chain joining a pair of points in a network.*

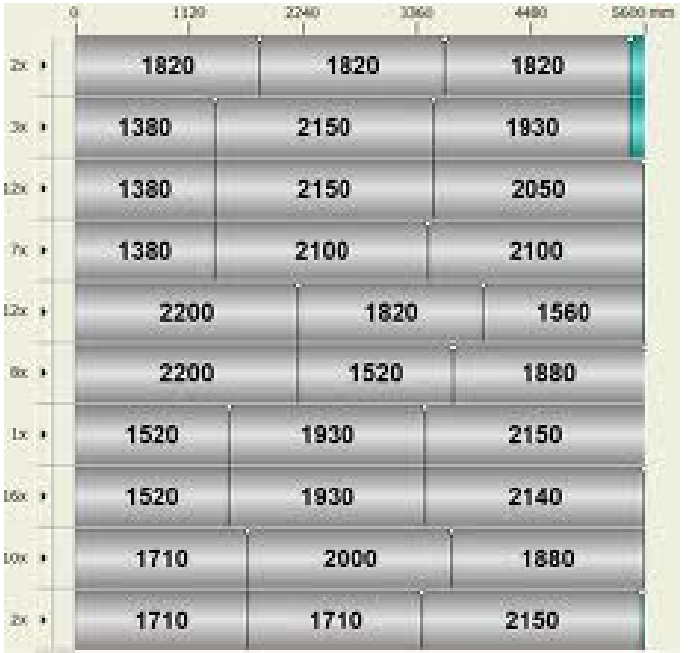
Motivation

Why should we deal with formulations that cannot fit in the memory or cannot be explicitly enumerated?

- Natural formulations: Cutting stock, Vehicle Routing, etc...
- Equivalent formulations: result of reformulation/decomposition principles

Cutting stock





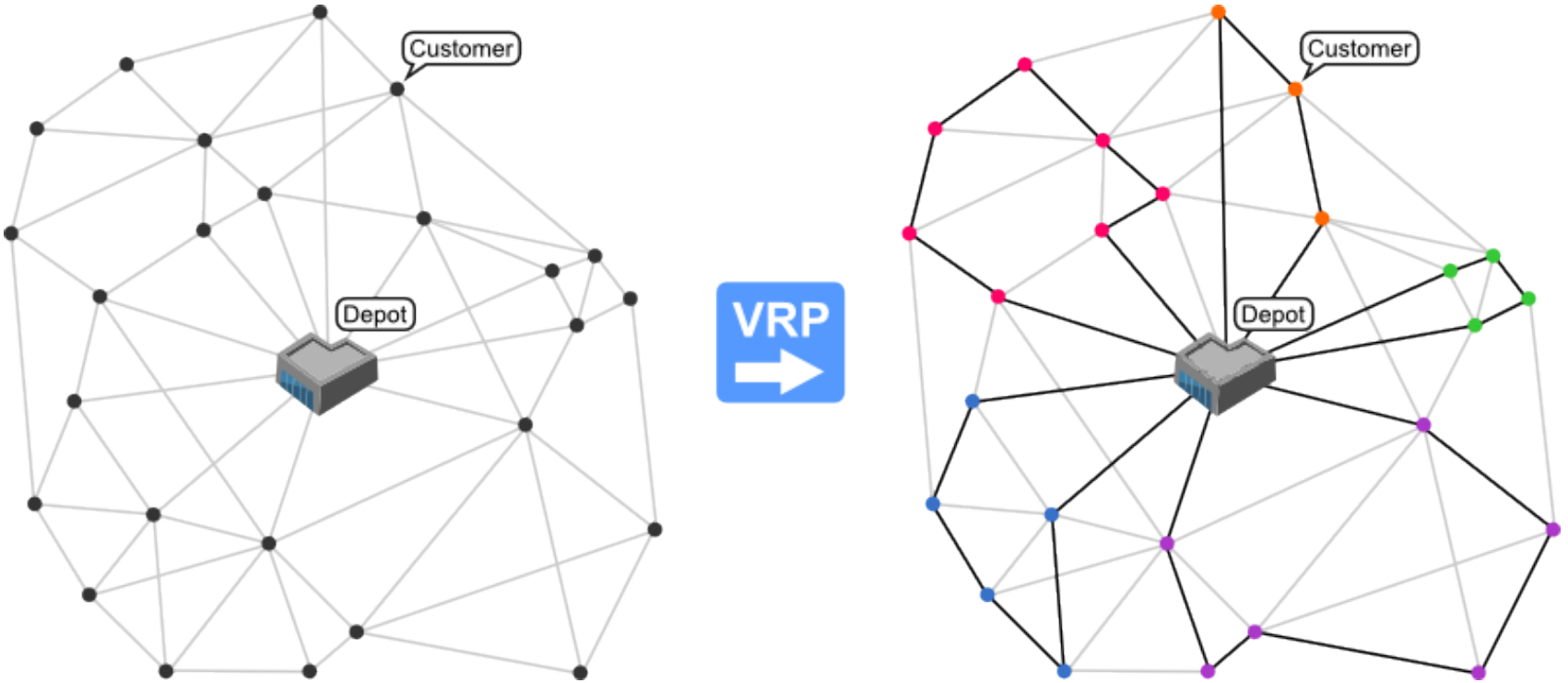
Satisfy a demand d_i of cuts of width w_i with the minimal number of rolls of size W.

Cutting stock

$$
\begin{aligned}
z = \min & \sum_{p \in P} x_p & \\
\text{s.t.} & \sum_{p \in P} a_{ip} x_p \geq d_i & \forall i \in I \\
& \mathbf{x} \in \mathbb{Z}_+^{|P|} &
\end{aligned}
$$

P is the set of feasible patterns, i.e., $\sum_{i \in p} a_{ip} w_i \leq W$. The number of possible patterns grows exponentially as a function of $|I|$.

Vehicle routing



Satisfy the delivery demand d_i of I customers with up to K vehicles of capacity Q minimizing the total traveling costs.

Vehicle routing

$$
\begin{aligned}
z = \min & \sum_{r \in R} c_r x_r & \\
\text{s.t.} & \sum_{r \in R} a_{ir} x_r \geq 1 & \forall i \in I \\
& \sum_{r \in R} x_r \leq |K| & \\
& \mathbf{x} \in \{0,1\}^{|R|} &
\end{aligned}
$$

R is the set of feasible routes, i.e., $\sum_{i \in r} a_{ir} d_i \leq Q$ with cost c_r. The number of routes grows exponentially as a function of $|I|$.

Equivalent formulations

George B. Dantzig; Philip Wolfe (1960). *Decomposition Principle for Linear Programs*. Operations Research 8: 101–111.

Informally, the core idea is to manipulate the formulation of a decision problem in order to obtain an alternative formulation with some nice properties:

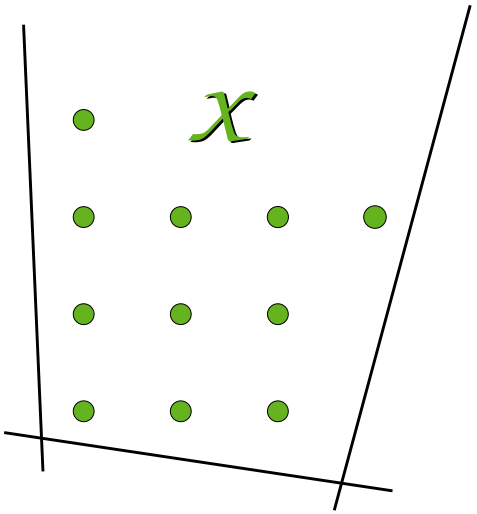
- Decomposes into several decision problems (Master and sub problems).
- Exhibits a stronger dual bound.

Under mild assumptions, decision problems with a natural "exponential" formulation can be viewed as a DW reformulation of a corresponding "compact" formulation.

Compact or Original formulation

We want to solve an integer program, the **compact** formulation

$$\begin{aligned} z \;=\; & \min \mathbf{c^t x} \\ & \text{s.t. } A\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \in X \end{aligned}$$



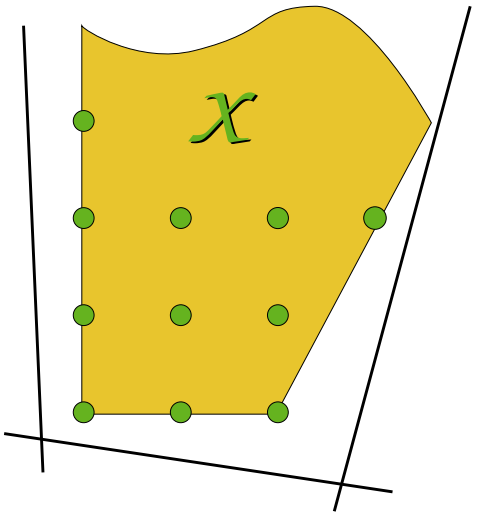
with $X = \{\mathbf{x} \in \mathbb{Z}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\} \neq \emptyset$

replacing X by $conv(X)$ does not change z^*

Compact or Original formulation

We want to solve an integer program, the **compact** formulation

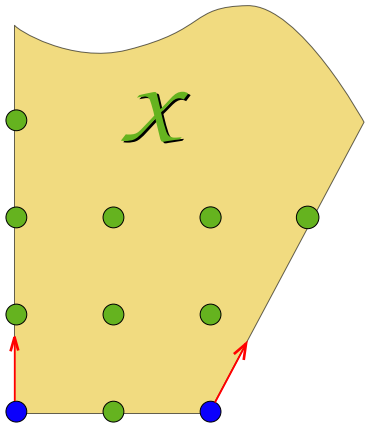
$$\begin{aligned} z \;=\; & \min \mathbf{c^t x} \\ & \text{s.t. } A\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \in X \end{aligned}$$



with $X = \{\mathbf{x} \in \mathbb{Z}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\} \neq \emptyset$
replacing X by $conv(X)$ does not change z^*

Minkowski-Weyl Theorem

Theorem: Every $\mathbf{x} \in conv(X)$ can be represented as



convex combination of **extreme points** $\{\mathbf{x_p}\} p \in P$ plus a non-negative combination of **extreme rays** $\{\mathbf{x_r}\} r \in R$ of $conv(X)$

$$\mathbf{x} = \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r, \quad \sum_{p \in P} \lambda_p = 1, \lambda \geq 0 \in \mathbb{R}^{|P|+|R|}$$

Applying the Dantzig-Wolfe decomposition

We transform the original formulation:

$$
\begin{aligned}
z \;=\; & \min \mathbf{c^t x} \\
& \text{s.t. } A\mathbf{x} \geq \mathbf{b} \\
& \mathbf{x} \in X = \{\mathbf{x} \in \mathbb{Z}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\}
\end{aligned}
$$

replacing the **external** representation for X by the **internal**

$$\sum_{p \in P} \lambda_p = 1$$

$$\lambda \geq 0$$

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

Applying the Dantzig-Wolfe decomposition

We transform the original formulation:

$$
\begin{aligned}
z \;=\; & \min \mathbf{c^t x} \\
& \text{s.t. } A\mathbf{x} \geq \mathbf{b} \\
& \quad \mathbf{x} \in X = \{\mathbf{x} \in \mathbb{Z}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\}
\end{aligned}
$$

replacing the **external** representation for X by the **internal**

$$
\begin{aligned}
& \sum_{p \in P} \lambda_p = 1 \\
& \lambda \geq 0 \\
& \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r = \mathbf{x}
\end{aligned}
$$

Applying the Dantzig-Wolfe decomposition

We transform the original formulation:

$$
\begin{aligned}
z \;=\; & \min \mathbf{c^t x} \\
& \text{s.t. } A\mathbf{x} \geq \mathbf{b} \\
& \qquad \sum_{p \in P} \lambda_p = 1 \\
& \qquad \lambda \geq 0 \\
& \qquad \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r = \mathbf{x}
\end{aligned}
$$

Applying the Dantzig-Wolfe decomposition

We substitute $\mathbf{x}$ by its equivalent representation:

$$
\begin{aligned}
z \;=\; \min\ & \mathbf{c^t}\Big(\sum_{p\in P} \mathbf{x_p}\lambda_p + \sum_{r\in R} \mathbf{x_r}\lambda_r\Big) \\
\text{s.t. } & A\Big(\sum_{p\in P} \mathbf{x_p}\lambda_p + \sum_{r\in R} \mathbf{x_r}\lambda_r\Big) \geq \mathbf{b} \\
& \sum_{p\in P} \lambda_p = 1 \\
& \lambda \geq 0 \\
& \sum_{p\in P} \mathbf{x_p}\lambda_p + \sum_{r\in R} \mathbf{x_r}\lambda_r = \mathbf{x}
\end{aligned}
$$

Applying the Dantzig-Wolfe decomposition

Finally, replace $c_j = \mathbf{c^t x_j}$ and $\mathbf{a_j} = A\mathbf{x_j}$

$$
\begin{aligned}
z \;=\; \min \; & \sum_{p \in P} \mathbf{c^t x_p} \lambda_p + \sum_{r \in R} \mathbf{c^t x_r} \lambda_r \\
\text{s.t.} \; & \sum_{p \in P} A\mathbf{x_p} \lambda_p + \sum_{r \in R} A\mathbf{x_r} \lambda_r \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \lambda \geq 0 \\
& \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r = \mathbf{x}
\end{aligned}
$$

Applying the Dantzig-Wolfe decomposition

Finally, replace $c_j = \mathbf{c^t x_j}$ and $\mathbf{a_j} = A\mathbf{x_j}$

$$
\begin{aligned}
z \;=\; \min \; & \sum_{p \in P} c_p \lambda_p + \sum_{r \in R} c_r \lambda_r \\
\text{s.t.} \; & \sum_{p \in P} \mathbf{a_p} \lambda_p + \sum_{r \in R} \mathbf{a_r} \lambda_r \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \lambda \geq 0 \\
& \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r = \mathbf{x}
\end{aligned}
$$

Extensive formulation

We keep the integrality on the **original** variables

$$
\begin{aligned}
z \;=\; \min \; & \sum_{p \in P} c_p \lambda_p + \sum_{r \in R} c_r \lambda_r \\
\text{s.t.} \; & \sum_{p \in P} \mathbf{a_p} \lambda_p + \sum_{r \in R} \mathbf{a_r} \lambda_r \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \boldsymbol{\lambda} \geq 0 \\
& \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r = \mathbf{x} \\
& \mathbf{x} \in \mathbb{Z}^n_+
\end{aligned}
$$

Extensive formulation

We keep the integrality on the **original** variables

$$
\begin{aligned}
z \;=\; \min \; & \sum_{p \in P} c_p \lambda_p + \sum_{r \in R} c_r \lambda_r \\
\text{s.t.} \; & \sum_{p \in P} \mathbf{a_p} \lambda_p + \sum_{r \in R} \mathbf{a_r} \lambda_r \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \boldsymbol{\lambda} \geq 0 \\
& \sum_{p \in P} \mathbf{x_p} \lambda_p + \sum_{r \in R} \mathbf{x_r} \lambda_r = \mathbf{x} \\
& \mathbf{x} \in \mathbb{Z}^n_+
\end{aligned}
$$

Keep in mind!

Integrality must hold for original **x** variables

Boundness

When the set $X = \{\mathbf{x} \in \mathbb{Z}^n_+ \mid D\mathbf{x} \geq \mathbf{d}\} \neq \emptyset$ is bounded, all points $\mathbf{x}$ can be represented as a convex combination of **extreme points** $\{\mathbf{x_p}\} p \in P$ of $conv(X)$

$$\mathbf{x} = \sum_{p \in P} \mathbf{x_p} \lambda_p, \ \sum_{p \in P} \lambda_p = 1, \lambda \geq 0 \in \mathbb{R}^{|P|}$$

from now on we focus on bounded polyhedra.

Two **equivalent** formulations

Compact formulation

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

Extensive formulation

$$
\begin{aligned}
z_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \mathbf{x} = \sum_{p \in P} \mathbf{x_p} \lambda_p \\
& \lambda \geq 0, \ \mathbf{x} \in \mathbb{Z}^n_+.
\end{aligned}
$$

Two **equivalent** formulations

Compact formulation

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

Extensive formulation

$$
\begin{aligned}
z_{EF} = \min \ & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \mathbf{x} = \sum_{p \in P} \mathbf{x_p} \lambda_p \\
& \lambda \geq 0, \ \mathbf{x} \in \mathbb{Z}^n_+.
\end{aligned}
$$

How do we solve them?

Two **equivalent** formulations

Compact formulation

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

Extensive formulation

$$
\begin{aligned}
z_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \mathbf{x} = \sum_{p \in P} \mathbf{x_p} \lambda_p \\
& \lambda \geq 0, \ \mathbf{x} \in \mathbb{Z}^n_+.
\end{aligned}
$$

How do we solve them?

by Branch and Bound (and Cut)

Relaxations z_{CF} and z_{EF}

Compact formulation

$$
\begin{aligned}
z_{CF} = \min \quad & \mathbf{c^t x} \\
\text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b} \\
& D\mathbf{x} \geq \mathbf{d} \\
& \mathbf{x} \geq 0.
\end{aligned}
$$

Extensive formulation

$$
\begin{aligned}
z_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \mathbf{x} = \sum_{p \in P} \mathbf{x_p} \lambda_p \\
& \lambda \geq 0, \ \mathbf{x} \geq 0.
\end{aligned}
$$

Relaxations z_{CF} and z_{EF}

Compact formulation

$$
\begin{aligned}
z_{CF} = \min \quad & \mathbf{c^t x} \\
\text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b} \\
& D\mathbf{x} \geq \mathbf{d} \\
& \mathbf{x} \geq 0.
\end{aligned}
$$

Extensive formulation

$$
\begin{aligned}
z_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \lambda \geq 0.
\end{aligned}
$$

Relaxations $\underline{z}_{CF}$ and $\underline{z}_{EF}$

Compact formulation

$$\begin{aligned} \underline{z}_{CF} = \min \quad & \mathbf{c^t x} \\ \text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b} \\ & D\mathbf{x} \geq \mathbf{d} \\ & \mathbf{x} \geq 0. \end{aligned}$$

Extensive formulation

$$\begin{aligned} \underline{z}_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\ \text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\ & \sum_{p \in P} \lambda_p = 1 \\ & \lambda \geq 0. \end{aligned}$$

$\underline{z}_{EF} \geq \underline{z}_{CF}$ (desirably $\underline{z}_{EF} > \underline{z}_{CF}$) as $\mathbf{x_p} \in conv(X)$ is an integer vector (not affected by relaxation)!
Unfortunately, nothing comes for free as $|P|$ grows exponentially with n.
P cannot be enumerated and we need **column genration**.

Master Problem: Linear Relaxation

We start with a subset of extreme points $\tilde{P}$:

$$\begin{aligned} z_{EF} \;=\; & \min \sum_{p \in \tilde{P}} c_p \lambda_p \\ & \text{s.t.} \sum_{p \in \tilde{P}} \mathbf{a_p} \lambda_p \geq \mathbf{b} && (\pi) \\ & \sum_{p \in \tilde{P}} \lambda_p = 1 && (\pi_0) \\ & \lambda \geq 0 \end{aligned}$$

Given $(z^*_{EF}, \pi^*, \pi^*_0)$, we search profitable columns:

$$rc = \min_{p \in P \setminus \tilde{P}} (\mathbf{c}^t - \pi^* A)\mathbf{x_p} - \pi^*_0$$

by implicit enumeration (solving an optimization problem):

$$rc = \min_{\mathbf{x} \in X} (\mathbf{c}^t - \pi^* A)\mathbf{x} - \pi^*_0$$

Pricing Problem

The so called *pricing problem* computes the smallest reduced cost of all variables :

$$\begin{aligned} rc \;=\; & \min\ (\mathbf{c}^t - \pi^* A)\mathbf{x} - \pi_0^* \\ & \text{s.t. } D\mathbf{x} \geq \mathbf{d} \\ & \mathbf{x} \in \mathbb{Z}_+^n. \end{aligned}$$

If $rc^* < 0$, then $\mathbf{x_p} = \mathbf{x}^*$ is added to $\tilde{P}$ (it is not in $\tilde{P}$ as $rc_{p \in \tilde{P}} \geq 0$) and the process iterates.
Otherwise, z^*_{EF} cannot be improved further and constitutes a valid bound to z^*.

Discretization

Another option for decomposition: instead of using $conv(X)$, X is directly reformulated.

When every integer $\mathbf{x} \in X$ can be written as an integral combination:

$$\mathbf{x} = \sum_{p \in P} \mathbf{x}_p \lambda_p + \sum_{r \in R} \mathbf{x}_r \lambda_r, \sum_{p \in P} \lambda_p = 1, \lambda \in \mathbb{Z}_+^{|P|+|R|}$$

When X is bounded, the reformulation leads to linear master problem even in case of non-linear objective function:

$$c(\mathbf{x}) = c\left(\sum_{p \in P} \mathbf{x}_p \lambda_p\right) = \sum_{p \in P} c_p \lambda_p$$

Illustration skip

Find the shortest path from 1 to 6

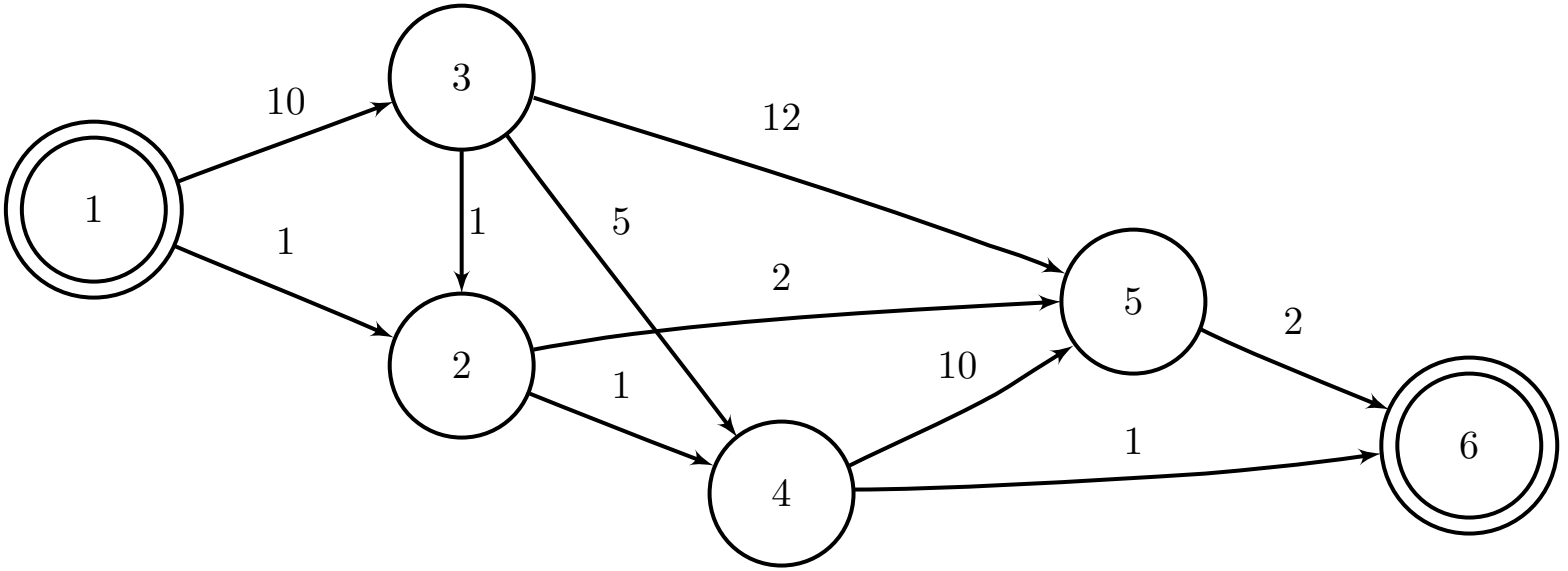
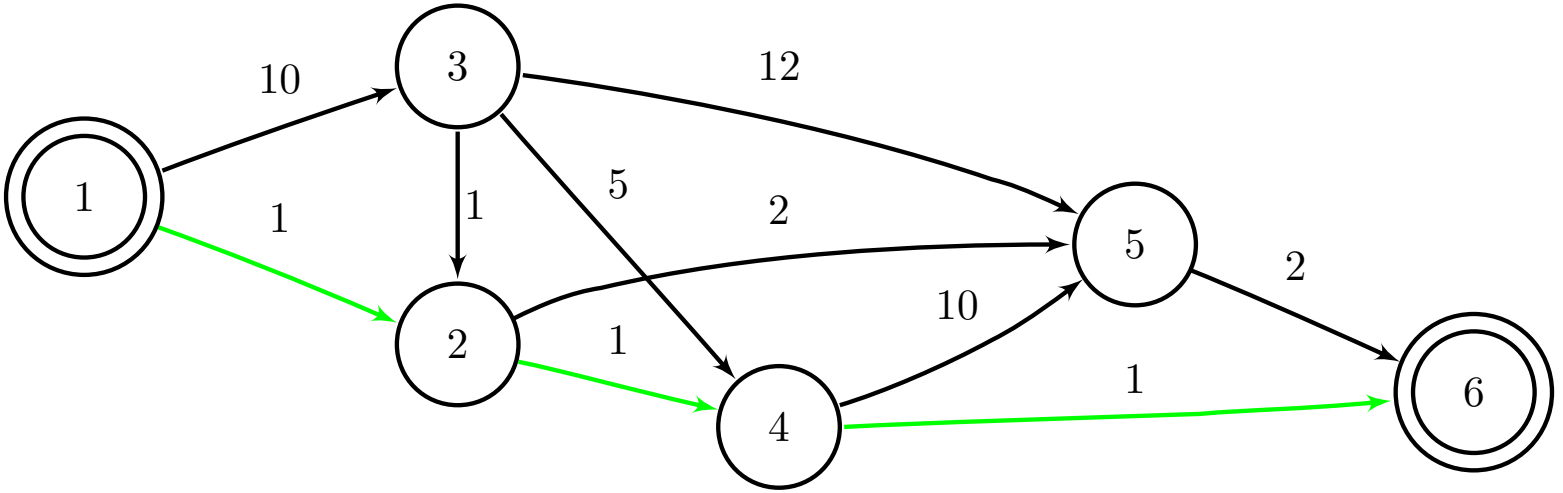


Illustration skip

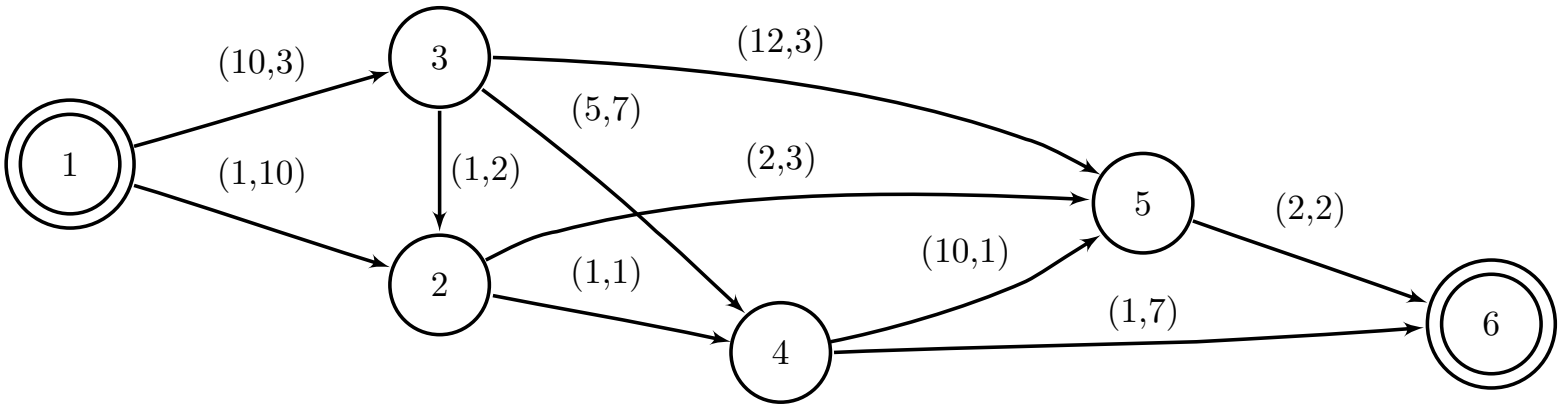
Find the shortest path from 1 to 6



Path 1-2-4-6 is optimal: cost 3

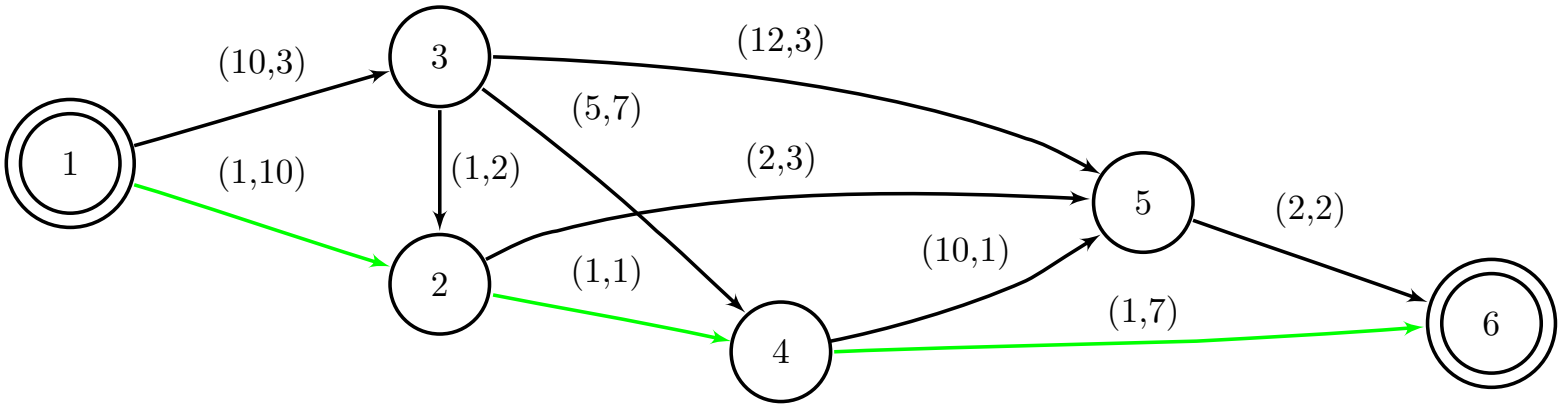
Example

Find the shortest path from 1 to 6
Total traversal time must not exceed 14 units



Example

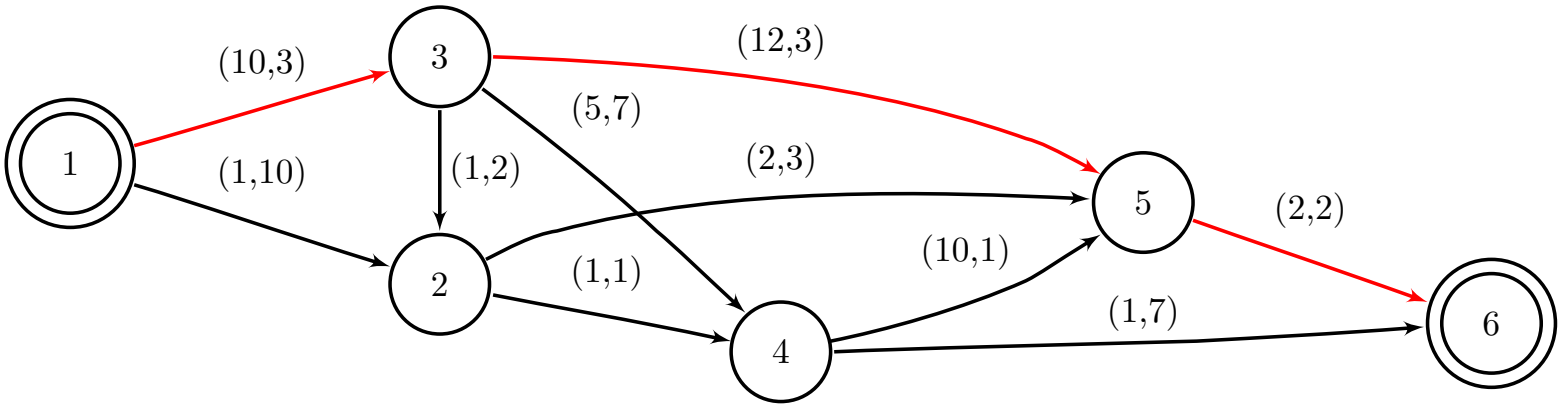
Find the shortest path from 1 to 6
Total traversal time must not exceed 14 units



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

Example

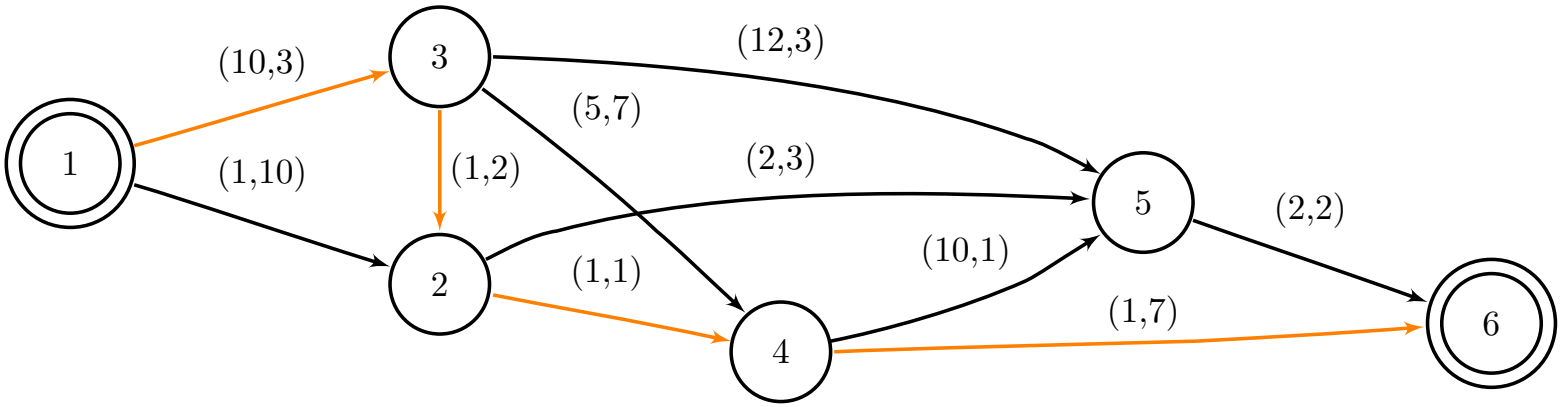
Find the shortest path from 1 to 6
Total traversal time must not exceed 14 units



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

Example

Find the shortest path from 1 to 6
Total traversal time must not exceed 14 units



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

Integer program (IP)

The problem is (weakly) NP-Hard, although pseudo-polynomial algorithms exist, we use an approach via Integer Programming.

Integer program (IP)

For each arc (i,j) the traversal cost and traversal time are given by c_{ij} and t_{ij}, respectively.

$$z = \texttt{min} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\texttt{s.t.} \sum_{j|(1,j)\in A} x_{1j} = 1$$

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i|(i,6)\in A} x_{i6} = 1$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leq 14$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Integer program (IP)

For each arc (i,j) the traversal cost and traversal time are given by c_{ij} and t_{ij}, respectively.

$$z = \texttt{min} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\texttt{s.t.} \sum_{j|(1,j)\in A} x_{1j} = 1$$

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i|(i,6)\in A} x_{i6} = 1$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leq 14$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Exploit embedded SPP

Path formulation

What remains

$$\sum_{j|(1,j)\in A} x_{1j} = 1$$

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i|(i,6)\in A} x_{i6} = 1$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

defines a network flow problem, which decomposes into flows on paths (and cycles)

Path formulation

The convex hull of

$$\sum_{j|(1,j)\in A} x_{1j} = 1$$

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i|(i,6)\in A} x_{i6} = 1$$

$$x_{ij} \geq 0 \qquad (i,j) \in A$$

defines a polyhedron with integer vertices

Path formulation

Every (fractional) flow can be represented as convex combination of paths (and cycles)

$$x_{ij} = \sum_{p \in P} x_{pij} \lambda_p \; (i,j) \in A$$

$$\sum_{p \in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \quad p \in P$$

P is the set of all paths from 1 to 6
$\sum_{p \in P} \lambda_p = 1$ is called convexity constraint

Master Problem

Let's substitute in (IP) x_{ij} with this representation

$$z = \texttt{min} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\texttt{s.t.} \sum_{j|(1,j)\in A} x_{1j} = 1$$

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i|(i,6)\in A} x_{i6} = 1$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leq 14$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Master Problem

Let's substitute in (IP) x_{ij} with this representation

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

Master Problem

Let's substitute in (IP) x_{ij} with this representation

$$z = \texttt{min} \sum_{(i,j)\in A} c_{ij} \sum_{p\in P} x_{pij}\lambda_p$$

$$\texttt{s.t.} \sum_{p\in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \qquad p \in P$$

$$\sum_{p\in P} x_{pij}\lambda_p = x_{ij} \qquad (i,j) \in A$$

$$\sum_{(i,j)\in A} t_{ij} \sum_{p\in P} x_{pij}\lambda_p \leq 14$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Master Problem

Let's substitute in (IP) x_{ij} with this representation

$$z = \texttt{min} \sum_{(i,j)\in A} c_{ij} \sum_{p\in P} x_{pij}\lambda_p$$

$$\texttt{s.t.} \sum_{(i,j)\in A} t_{ij} \sum_{p\in P} x_{pij}\lambda_p \leq 14$$

$$\sum_{p\in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \qquad p \in P$$

$$\sum_{p\in P} x_{pij}\lambda_p = x_{ij} \qquad (i,j) \in A$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Master Problem

Let's substitute in (IP) x_{ij} with this representation

$$z = \texttt{min} \sum_{p \in P} \Big(\sum_{(i,j) \in A} c_{ij} x_{pij} \Big) \lambda_p$$

$$\texttt{s.t.} \sum_{p \in P} \Big(\sum_{(i,j) \in A} t_{ij} x_{pij} \Big) \lambda_p \leq 14$$

$$\sum_{p \in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \qquad p \in P$$

$$\sum_{p \in P} x_{pij} \lambda_p = x_{ij} \qquad (i,j) \in A$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Master Problem

Let's substitute in (IP) x_{ij} with this representation

$$z = \texttt{min} \sum_{p \in P} \Big(\sum_{(i,j) \in A} c_{ij} x_{pij} \Big) \lambda_p$$

Keep in mind!

Integrality must hold for original x_{ij} variables

$$\sum_{p \in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \qquad p \in P$$

$$\sum_{p \in P} x_{pij} \lambda_p = x_{ij} \qquad (i,j) \in A$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A$$

Master Problem

Relax integrality constraint on x_{ij}

$$z = \texttt{min} \sum_{p \in P} \Big(\sum_{(i,j) \in A} c_{ij} x_{pij} \Big) \lambda_p$$

$$\texttt{s.t.} \sum_{p \in P} \Big(\sum_{(i,j) \in A} t_{ij} x_{pij} \Big) \lambda_p \leq 14$$

$$\sum_{p \in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \qquad p \in P$$

$$\sum_{p \in P} x_{pij} \lambda_p = x_{ij} \qquad (i,j) \in A$$

$$x_{ij} \geq 0 \qquad (i,j) \in A$$

Master Problem

Remove the linking constraints between λ_p and x_{ij}

$$z = \texttt{min} \sum_{p \in P} \Big(\sum_{(i,j) \in A} c_{ij} x_{pij} \Big) \lambda_p$$

$$\texttt{s.t.} \sum_{p \in P} \Big(\sum_{(i,j) \in A} t_{ij} x_{pij} \Big) \lambda_p \leq 14$$

$$\sum_{p \in P} \lambda_p = 1$$

$$\lambda_p \geq 0 \qquad p \in P$$

Problem: in this small example $|P| = 9$, in general we have a lot of paths.

Master Problem

The whole formulation

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

Master Problem

We work with a subset of the variables

$$\begin{aligned} \texttt{min}\ & 14\lambda_{12456} + 5\lambda_{1256} \\ \texttt{s.t.}\ & 14\lambda_{12456} + 15\lambda_{1256} \leq 14 \\ & \lambda_{12456} + \lambda_{1256} = 1 \\ & \lambda_{12456} \ , \ \lambda_{1256} \geq 0 \end{aligned}$$

called restricted master problem (RMP)

Restricted Master Problem (RMP)

How to generate such a new column we don't have?

$$\begin{aligned}
\texttt{min} \ldots + 24\lambda_{132456} + \ldots & \\
\texttt{s.t.} \ldots + 9\lambda_{132456} + \ldots &\leq 14 \\
\ldots + \lambda_{132456} + \ldots &= 1 \\
\ldots \ , \quad \lambda_{132456} + \ldots &\geq 0
\end{aligned}$$

Restricted Master Problem (RMP)

How to generate such a new column we don't have?

$$
\begin{array}{rrcl}
 & & & \texttt{duals} \\
 & & & \downarrow \\
\texttt{min} & \ldots + 24\lambda_{132456} + \ldots & & \\
\texttt{s.t.} & \ldots + 9\lambda_{132456} + \ldots & \leq 14 & \pi_1 \\
 & \ldots + \lambda_{132456} + \ldots & = 1 & \pi_0 \\
 & \ldots \quad , \quad \lambda_{132456} + \ldots & \geq 0 &
\end{array}
$$

Simplex method: to enter the basis a variable must have a negative reduced cost:

$$
\begin{aligned}
\overline{c}_{132456} &= 24 - (\pi_1, \pi_0) \cdot \begin{pmatrix} 9 \\ 1 \end{pmatrix} \\
&= 24 - 9\pi_1 - 1\pi_0 < 0
\end{aligned}
$$

Pricing subproblem

Checking whether such λ_p variable exists, with

$$\overline{c}_p = \sum_{(i,j)\in A} c_{ij} x_{pij} - \Big(\sum_{(i,j)\in A} t_{ij} x_{pij} \Big) \pi_1 - \pi_0 < 0$$

is an optimization problem called Pricing Subproblem:

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

s.t. x_{ij} encodes a feasible column, that is

$$\text{s.t.} \sum_{j|(1,j)\in A} x_{1j} = 1$$

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i|(i,6)\in A} x_{i6} = 1$$

$$x_{ij} \geq 0 \qquad (i,j) \in A$$

Pricing subproblem

Checking whether such λ_p variable exists, with

$$\overline{c}_p = \sum_{(i,j)\in A} c_{ij} x_{pij} - \Big(\sum_{(i,j)\in A} t_{ij} x_{pij} \Big) \pi_1 - \pi_0 < 0$$

is an optimization problem called Pricing Subproblem:

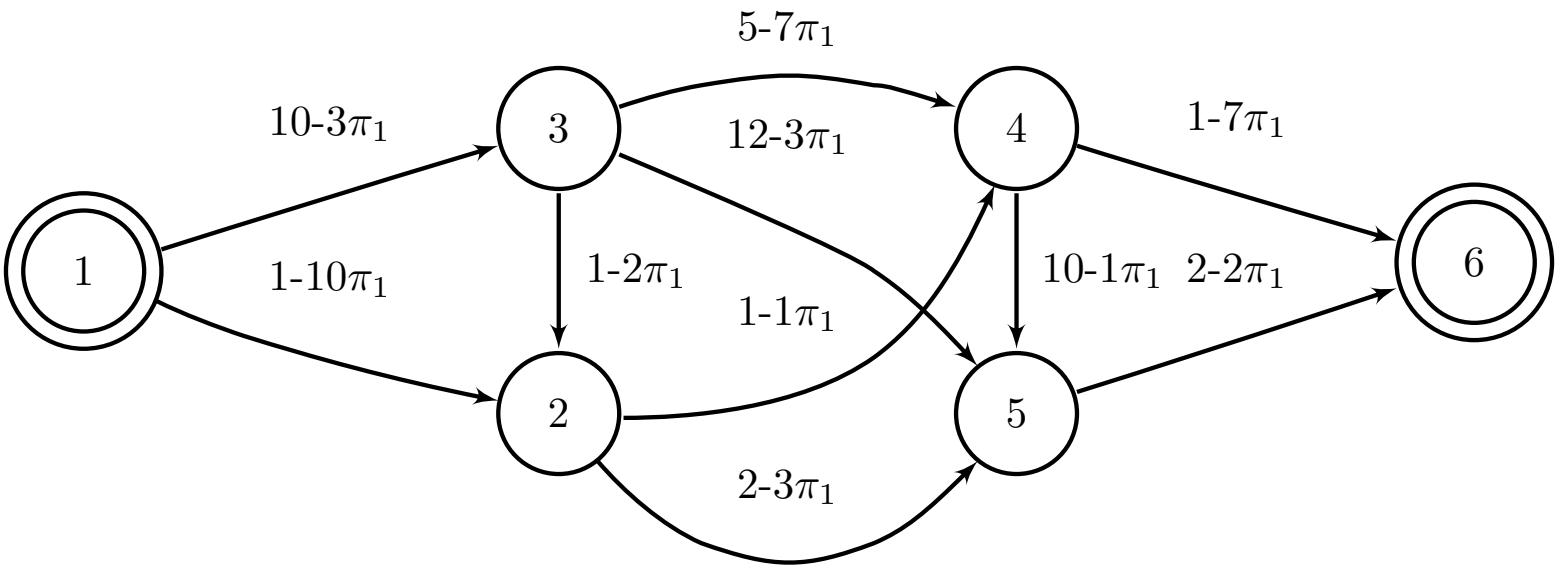
$$\overline{c}^* = \texttt{min} \sum_{(i,j)\in A} (c_{ij} - \pi_1 t_{ij}) x_{ij} - \pi_0$$

s.t. x_{ij} encodes a feasible column, that is

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

Pricing subproblem

In our case:



What we have done in practice

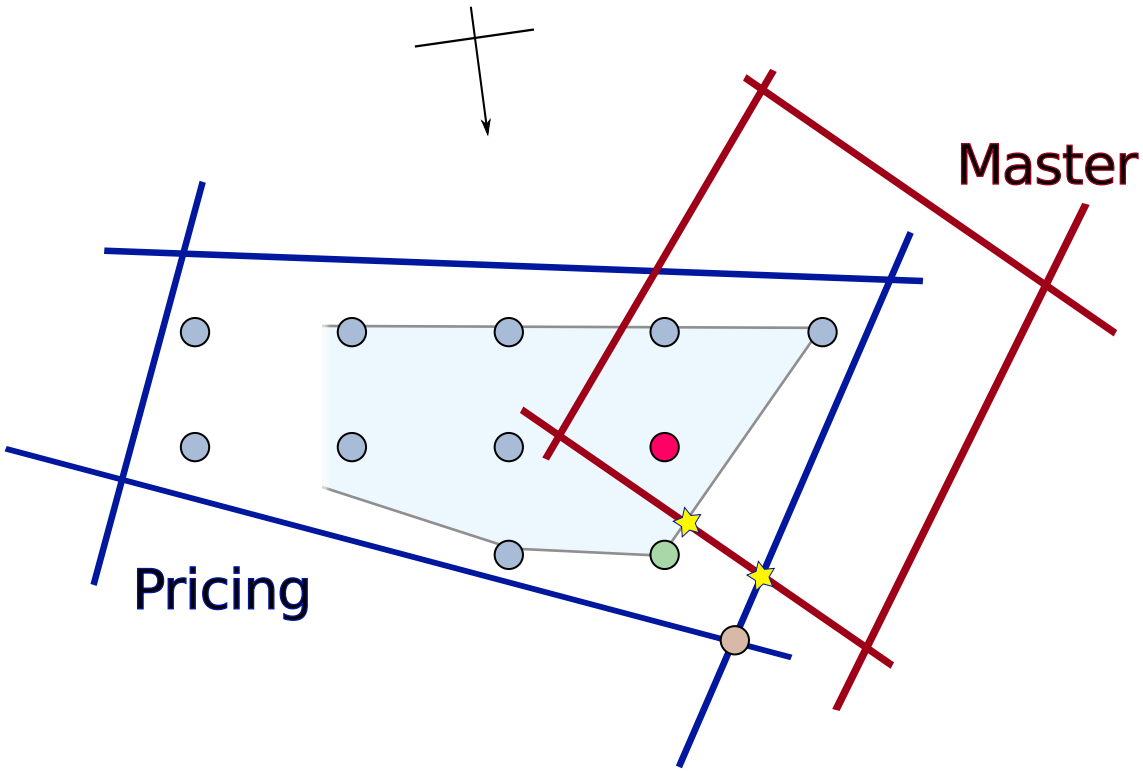
- Formulate a (simple) combinatorial problem
- Recognize a known (possibly easier) subproblem
- Reformulate the original problem in terms of Master and Pricing subproblems

Remarks:

- Huge number of variable
- Modified cost for the easy subproblem

Integrality property - does not hold

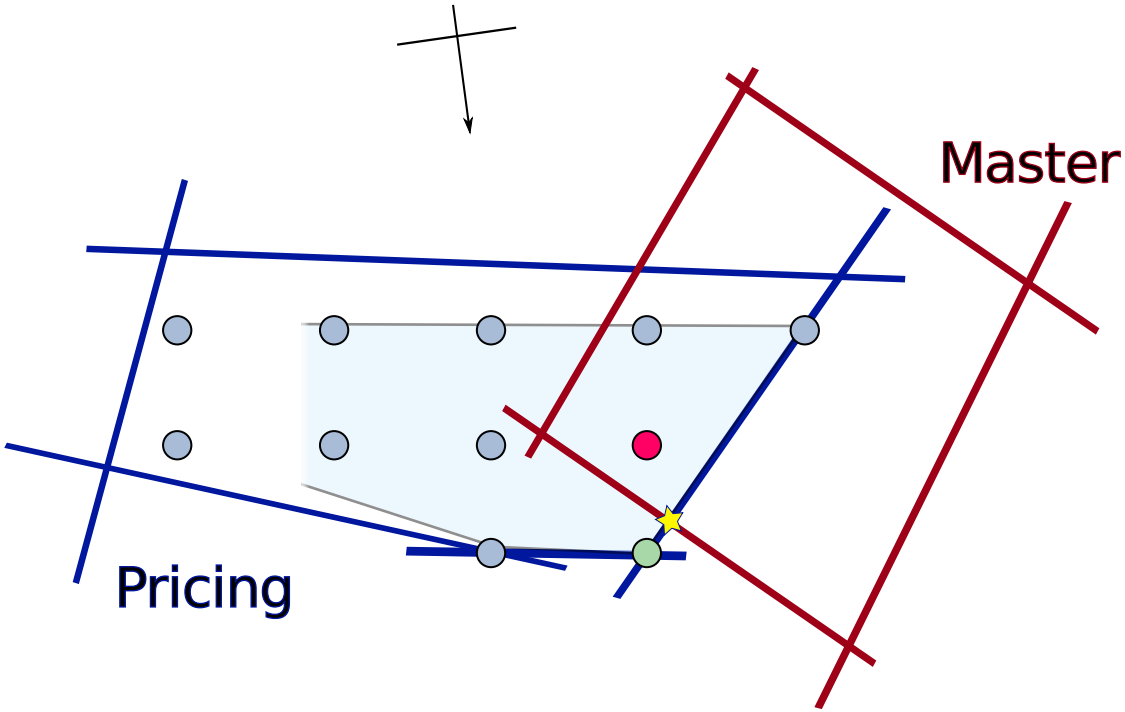
The lower bound obtained from the RMP (z_{EF}) is as weak as the original LP relaxation when the subproblem is solved as an LP



The integrality gap may be closed partly when the subproblem is solved as an IP

Integrality property - holds

But precisely this opportunity is not present when the integrality property holds



An easier pricing problem may imply a weaker LP relaxation of the RMP

Which $D\mathbf{x} \geq \mathbf{d}$? - Practical advice

Some general rules:

- Look for structures for which you have a good algorithm in your arsenal. E.g., knapsack, shortest paths.
- I tend to move complexity to the pricing:
  - The master problem is the only problem that is 'relaxed' for its solution: if everything is in the master the decompositon is not going to be beneficial.
  - For a well structured pricing problem (e.g., resource constrained shortest paths) powerful heuristics and effective relaxations can help to tackle the complexity.

Example, solving the LRMP skip

$$
\begin{array}{lrlll}
\overline{z} = \texttt{min} & 100\lambda_0 & & & \\
\texttt{s.t.} & 0\lambda_0 & & \leq 14 & \pi_1 \\
& \lambda_0 & & = 1 & \pi_0 \\
& \lambda_0 & & \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p

Example, solving the LRMP skip

$$
\begin{array}{lrll}
\overline{z} = \texttt{min}\ 100\lambda_0 & & & \\
\quad \texttt{s.t.} \quad 0\lambda_0 & & \leq 14 & \pi_1 \\
\qquad\qquad \lambda_0 & & = \ 1 & \pi_0 \\
\qquad\qquad \lambda_0 & & \geq \ 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00				

Example, solving the LRMP skip

$$
\begin{aligned}
\overline{z} = \min\ & 100\lambda_0 & & & \\
\text{s.t.}\quad & 0\lambda_0 & & \leq 14 & \pi_1 \\
& \lambda_0 & & = 1 & \pi_0 \\
& \lambda_0 & & \geq 0 &
\end{aligned}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18

Example, solving the LRMP skip

$$
\begin{array}{lrcl}
\overline{z} = \texttt{min} & 100\lambda_0 + 3\lambda_{1246} & & \\
\texttt{s.t.} & 0\lambda_0 + 18\lambda_{1246} & \leq 14 & \pi_1 \\
& \lambda_0 + \lambda_{1246} & = 1 & \pi_0 \\
& \lambda_0 \ , \ \lambda_{1246} & \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18

Example, solving the LRMP skip

$$
\begin{array}{lrcl}
\overline{z} = \texttt{min} & 100\lambda_0 + 3\lambda_{1246} & & \\
\texttt{s.t.} & 0\lambda_0 + 18\lambda_{1246} & \leq 14 & \pi_1 \\
& \lambda_0 + \lambda_{1246} & = 1 & \pi_0 \\
& \lambda_0 \quad , \quad \lambda_{1246} & \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39				

Example, solving the LRMP skip

$$
\begin{array}{lrcl}
\overline{z} = \texttt{min} & 100\lambda_0 + 3\lambda_{1246} & & \\
\texttt{s.t.} & 0\lambda_0 + 18\lambda_{1246} & \leq 14 & \pi_1 \\
& \lambda_0 + \lambda_{1246} & = 1 & \pi_0 \\
& \lambda_0 \ , \ \lambda_{1246} & \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8

Example, solving the LRMP skip

$$
\begin{array}{lrcrcrcll}
\overline{z} = \texttt{min} & 100\lambda_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & & & \\
\texttt{s.t.} & 0\lambda_0 & + & 18\lambda_{1246} & + & 8\lambda_{1356} & \leq & 14 & \pi_1 \\
 & \lambda_0 & + & \lambda_{1246} & + & \lambda_{1356} & = & 1 & \pi_0 \\
 & \lambda_0 & , & \lambda_{1246} & , & \lambda_{1356} & \geq & 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8

Example, solving the LRMP skip

$$
\begin{array}{lrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 + & 3\lambda_{1246} + & 24\lambda_{1356} & & & & \\
\texttt{s.t.} & 0\lambda_0 + & 18\lambda_{1246} + & 8\lambda_{1356} & & \leq & 14 & \pi_1 \\
& \lambda_0 + & \lambda_{1246} + & \lambda_{1356} & & = & 1 & \pi_0 \\
& \lambda_0 \quad , & \lambda_{1246} \quad , & \lambda_{1356} & & \geq & 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10				

Example, solving the LRMP skip

$$
\begin{array}{lrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 + & 3\lambda_{1246} + & 24\lambda_{1356} & & & & \\
\texttt{s.t.} & 0\lambda_0 + & 18\lambda_{1246} + & 8\lambda_{1356} & & \leq & 14 & \pi_1 \\
& \lambda_0 + & \lambda_{1246} + & \lambda_{1356} & & = & 1 & \pi_0 \\
& \lambda_0 \quad , & \lambda_{1246} \quad , & \lambda_{1356} & & \geq & 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10

Example, solving the LRMP skip

$$
\begin{array}{rrrrrrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & & & \\
\texttt{s.t.} & 0\lambda_0 & + & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & & \leq 14 & \pi_1 \\
 & \lambda_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & & = 1 & \pi_0 \\
 & \lambda_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & & \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10

Example, solving the LRMP skip

$$
\begin{array}{lrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 + & 3\lambda_{1246} + & 24\lambda_{1356} + & 15\lambda_{13256} & & & \\
\texttt{s.t.} & 0\lambda_0 + & 18\lambda_{1246} + & 8\lambda_{1356} + & 10\lambda_{13256} & \leq & 14 & \pi_1 \\
& \lambda_0 + & \lambda_{1246} + & \lambda_{1356} + & \lambda_{13256} & = & 1 & \pi_0 \\
& \lambda_0 \ , & \lambda_{1246} \ , & \lambda_{1356} \ , & \lambda_{13256} & \geq & 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	−1.50				

Example, solving the LRMP skip

$$
\begin{array}{rrrrrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & & \\
\texttt{s.t.} & 0\lambda_0 & + & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & \leq 14 & \pi_1 \\
& \lambda_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & = 1 & \pi_0 \\
& \lambda_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	−1.50	−2.5	1256	5	15

Example, solving the LRMP skip

$$
\begin{array}{lrrrrrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & \\
\texttt{s.t.} & 0\lambda_0 & + & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} \leq 14 & \pi_1 \\
& \lambda_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} = 1 & \pi_0 \\
& \lambda_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	−1.50	−2.5	1256	5	15

Example, solving the LRMP skip

$$
\begin{array}{lrrrrrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & \\
\texttt{s.t.} & 0\lambda_0 & + & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} \leq 14 & \pi_1 \\
& \lambda_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} = 1 & \pi_0 \\
& \lambda_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} \geq 0 &
\end{array}
$$

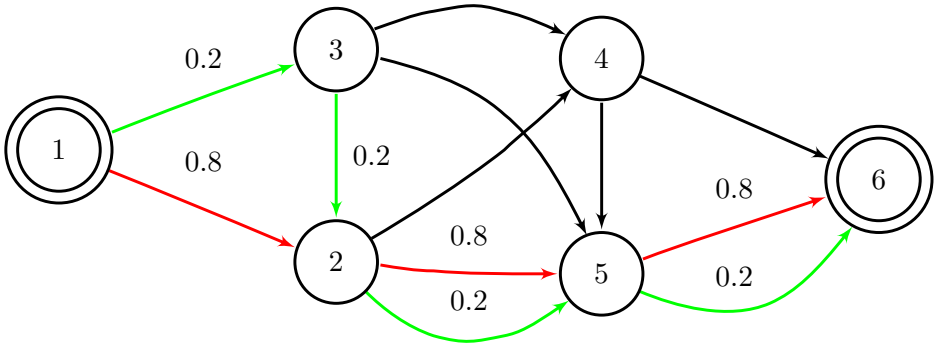
Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	−1.50	−2.5	1256	5	15
$\lambda_{1256} = 0.8, \lambda_{13256} = 0.2$	7.0	35.00	−2.00				

Example, solving the LRMP skip

$$
\begin{array}{lrrrrrrrrrrl}
\overline{z} = \texttt{min} & 100\lambda_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & \\
\texttt{s.t.} & 0\lambda_0 & + & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} \leq 14 & \pi_1 \\
& \lambda_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} = 1 & \pi_0 \\
& \lambda_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} \geq 0 &
\end{array}
$$

Master Solution	$\overline{z}$	π_0	π_1	$\overline{c}^*$	p	c_p	t_p
$\lambda_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18
$\lambda_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	−5.39	−32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	−2.10	−4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	−1.50	−2.5	1256	5	15
$\lambda_{1256} = 0.8, \lambda_{13256} = 0.2$	7.0	35.00	−2.00	0			

Example, solving the LRMP



Remarks:

- We use 0.2 times path 13256
- We use 0.8 times path 1256, which is infeasible
- A lower bound on the optimal solution is 7.0

Integer solutions

We apply Branch& Bound to obtain an integer solution.
In every node of the search tree we need to

- apply the reformulation
- solve the RMP by column generation

Branching on extensive formulation variables

We have two variables $\lambda_{13256} = 0.2$ and $\lambda_{1256} = 0.8$. We branch on one of them.

This is usually a bad idea

Branch on $\lambda_{13256} = 1$ is nice: it reduces the problem and improve the LB

Branch on $\lambda_{13256} = 0$ is bad: it just forbids this path!

In most of the cases the LB doesn't change, moreover you have to forbid its generation in the pricing (it is the most profitable path for the pricing since it was in the basis)

Nobody asked for the integrality of lambda variables.

Branching on extensive formulation variables

We have two variables $\lambda_{13256} = 0.2$ and $\lambda_{1256} = 0.8$. We branch on one of them.

This is usually a bad idea

Branch on $\lambda_{13256} = 1$ is nice: it reduces the problem and improve the LB

Branch on $\lambda_{13256} = 0$ is bad: it just forbids this path!

In most of the cases the LB doesn't change, moreover you have to forbid its generation in the pricing (it is the most profitable path for the pricing since it was in the basis)

Nobody asked for the integrality of lambda variables.

Branching on extensive formulation variables

We have two variables $\lambda_{13256} = 0.2$ and $\lambda_{1256} = 0.8$. We branch on one of them.

This is usually a bad idea

Branch on $\lambda_{13256} = 1$ is nice: it reduces the problem and improve the LB

Branch on $\lambda_{13256} = 0$ is bad: it just forbids this path!

In most of the cases the LB doesn't change, moreover you have to forbid its generation in the pricing (it is the most profitable path for the pricing since it was in the basis)

Nobody asked for the integrality of lambda variables.

Branching on extensive formulation variables

We have two variables $\lambda_{13256} = 0.2$ and $\lambda_{1256} = 0.8$. We branch on one of them.
This is usually a bad idea
Branch on $\lambda_{13256} = 1$ is nice: it reduces the problem and improve the LB
Branch on $\lambda_{13256} = 0$ is bad: it just forbids this path!
In most of the cases the LB doesn't change, moreover you have to forbid its generation in the pricing (it is the most profitable path for the pricing since it was in the basis)

Nobody asked for the integrality of lambda variables.

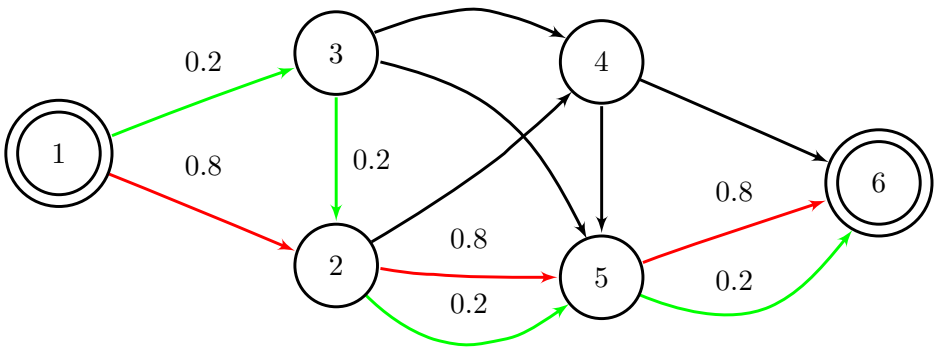
Branching on extensive formulation variables

We have two variables $\lambda_{13256} = 0.2$ and $\lambda_{1256} = 0.8$. We branch on one of them.
This is usually a bad idea
Branch on $\lambda_{13256} = 1$ is nice: it reduces the problem and improve the LB
Branch on $\lambda_{13256} = 0$ is bad: it just forbids this path!
In most of the cases the LB doesn't change, moreover you have to forbid its generation in the pricing (it is the most profitable path for the pricing since it was in the basis)

Nobody asked for the integrality of lambda variables.

Branching on compact formulation variables

Recall the compact-extensive formulations pair. The decomposition permits to reconstruct a solution of the compact formulation using the solution of the extensive formulation
In our example the compact formulation has one flow variable for each arc (for example $x_{12} = 0.8$)



Branching on compact formulation variables

By fixing $x_{12} = 0$ we remove arc $(1,2)$ from the network, by consequence variables λ_{1246} and λ_{1256} have to be removed from RMP
By fixing $x_{12} = 1$ we remove arc $(1,3)$ from the network, by consequence variables λ_{1356} and λ_{13256} have to be removed
Remark: We have to compute a valid LB for each node by column generation.

Nice things

Search tree is balanced and the pricing problem structure is unchanged

Branching on compact formulation variables

By fixing $x_{12} = 0$ we remove arc $(1,2)$ from the network, by consequence variables λ_{1246} and λ_{1256} have to be removed from RMP
By fixing $x_{12} = 1$ we remove arc $(1,3)$ from the network, by consequence variables λ_{1356} and λ_{13256} have to be removed
Remark: We have to compute a valid LB for each node by column generation.

Nice things

Search tree is balanced and the pricing problem structure is unchanged

Branching on compact formulation variables

By fixing $x_{12} = 0$ we remove arc $(1,2)$ from the network, by consequence variables λ_{1246} and λ_{1256} have to be removed from RMP
By fixing $x_{12} = 1$ we remove arc $(1,3)$ from the network, by consequence variables λ_{1356} and λ_{13256} have to be removed
Remark: We have to compute a valid LB for each node by column generation.

Nice things

Search tree is balanced and the pricing problem structure is unchanged

Branching options

If you are stuck in devising the branching scheme:

- Branch on original variables: when all pricing are distinct or just one pricing (recommend to handle it in the pricing).
- Branch on original variables, lexicographic order of original variables: useful for symmetric subproblems.
- Branch on aggregate original variables: when possible, useful for symmetric subproblems, may affect the pricing.
- Branch on auxiliary variables: when previous rules fails, extended original formulation.
- Nested partition of convexity constraints: select a subset of columns sharing common bounds on original variables $\sum_{p \in P: x_{pi} \geq l_i} \lambda_p = \delta \notin \mathbb{Z}_+$.

Branching conclusions

Branching on master variables is sometimes not feasible: nobody asks for integer master variables!
Not advisable: a master variable to zero has no effect on the bound, produces an unbalanced search tree and it requires to not generate a particular solution to the pricing problem (certainly with negative reduced cost).

Valid inequalities

Formulations can be strengthened by valid inequalities. Currently, we have **two** formulations that can be exploited.

Valid inequalities for the compact formulation (expressed in terms of x variables)

$$F\mathbf{x} \geq \mathbf{f}$$

Valid inequalities for the extensive formulation (expressed in terms of λ variables)

$$G\lambda \geq \mathbf{g}$$

Valid inequalities - Original Variables

Assume $F\mathbf{x} \geq \mathbf{f}$ are valid for the compact formulation, via reformulation:

$$\sum_{p \in P} \mathbf{f}_p \lambda_p + \sum_{r \in R} \mathbf{f}_r \lambda_r \geq \mathbf{f}$$

The duals α only affect the objective function of the pricing:

$$\min_{\mathbf{x} \in X} \ \mathbf{c}\mathbf{x} - \pi A\mathbf{x} - \alpha F\mathbf{x} - \pi_0$$

Or considering implicitly the cut in the pricing $X_F = \{\mathbf{x} \in X | F\mathbf{x} \geq \mathbf{f}\}$

$$\min_{\mathbf{x} \in X_F} \ \mathbf{c}\mathbf{x} - \pi A\mathbf{x} - \pi_0$$

Generic cutting planes (e.g. Chvatal-Gomory cuts) formulated on the original x variables have little or no effect.
Problem specific valid inequalities are the alternative to go. In particular, focus on the part of the problem not subject to convexification.

Valid inequalities - Master Variables

Some valid inequalites cannot be formulated in terms of original variables (e.g., k-path inequalities, subset-row inequalities)
Assume $G\lambda \geq \mathbf{g}$ are valid for the extensive formulation:

$$\sum_{p \in P} \mathbf{g}_p \lambda_p + \sum_{r \in R} \mathbf{g}_r \lambda_r \geq \mathbf{g}$$

The duals β must be considered and we assume we are able to write the coefficient $\mathbf{g}_j = g(\mathbf{a_j}) = g(A\mathbf{x})$:

$$\min_{\mathbf{x} \in X} \ \mathbf{cx} - \pi A\mathbf{x} - \beta g(A\mathbf{x}) - \pi_0$$

$g(A\mathbf{x})$ can be non-linear and possibly complicating for the subproblem.
If g is linear we can use cuts on original variables: $g(A\mathbf{x}) = F\mathbf{x}$. To derive a practical framework it is useful to consider new variables $\mathbf{y} = g(A\mathbf{x})$.

Practical implementations

A set of personal thoughts.
Frameworks for BCP:

- ABACUS: framework, C++, not well documented.
- **BCP**: framework, C++, well documented and stable, examples, tricky when comes to branching.
- **SCIP**: solver+framework, C++, flexible, branching more intuitive.
- DIP (CoinOR, DECOMP): framework, C++ and python, early stages of development.

Generic MIP based on branch-and-cut-and-price:

- **SYMPHONY**: solver for MILP, C, well documented, not fully conceived for user extension.
- BaPCod
- GCG (SCIP as a generic solver based on decomposition)

Some informal advices for your own implementation.

Practical implementations

Cutting stock problem by column generation.
Master:

$$
\begin{aligned}
z_{EF} = \min & \sum_{p \in P} \lambda_p \\
\text{s.t.} & \sum_{p \in P} a_{ip} \lambda_p \geq d_i \qquad \forall i \in I \\
& \lambda \in \mathbb{Z}_+^{|P|}
\end{aligned}
$$

Pricing:

$$
\begin{aligned}
rc = \min & \ (1 - \sum_{i \in I} \pi_i a_i) = 1 - \max \ \sum_{i \in I} \pi_i a_i \\
\text{s.t.} & \sum_{i \in I} w_i a_i \leq W \\
& \mathbf{a} \in \mathbb{Z}_+^{|I|}
\end{aligned}
$$

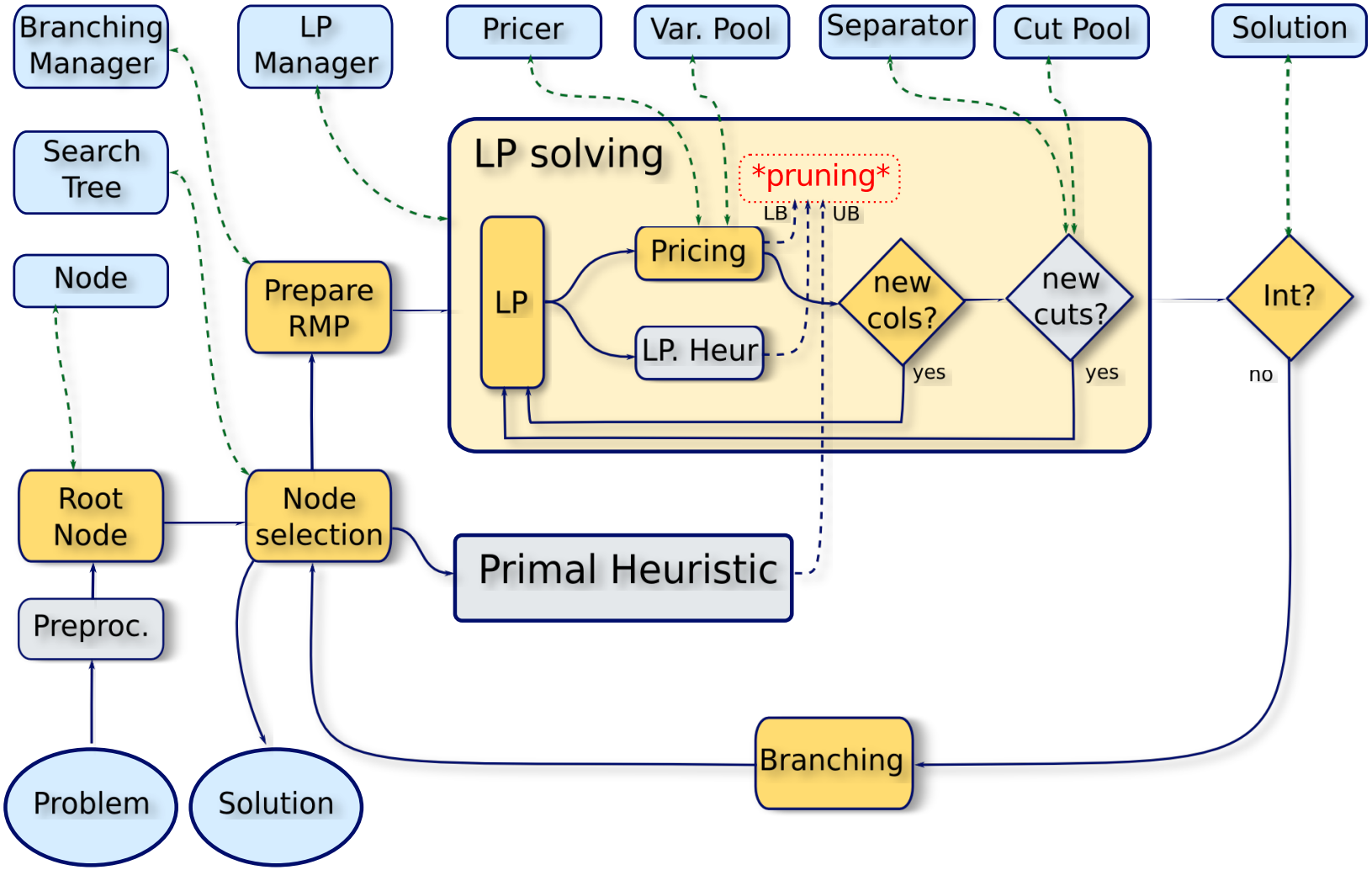
Check provided implementation in cutstock_gurobi.py

Exercise

Cutting stock alternative (compact) formulation:

$$
\begin{aligned}
z_{CF} = \min \ & \sum_k y_k & \\
\text{s.t.} \ & \sum_k x_k^i = d_i & \forall i \in I \\
& \sum_{i \in I} w_i x_k^i \leq W \cdot y_k & \forall k \in K \\
& y_k \in \{0,1\}, \mathbf{x} \in \mathbb{Z}_+^{|I| \times |K|} &
\end{aligned}
$$

Branch and Price and Cut



Other best practices

For branching and cutting plane purposes, original x values are necessary. Two alternatives:

- Keep linking constraints in the master problem:

$$\mathbf{x} = \sum_{p \in P} \mathbf{x}_p \lambda_p$$

- Work with $\mathbf{x}^*$ implicitly.

Other best practices

Suggested speedup techniques:

- Pricing problem (in particular for formulations with hard pricing problems):
  - Use a column pool.
  - Use local search over basic variables.
  - Use pricing heuristics.
  - When everything fails, switch to a relaxed pricing and hope that the dual gap is sufficient to stop.
- Master problem (in particular for formulations with degeneracy or a lot of columns):
  - Use stabilization (see next).
  - Consider lagrangean relaxation (when simplex is too heavy)
  - Use primal heuristics hoping for early pruning
  - Consider early termination of column generation
- Consider cutting planes, pricing or cutting first?

Lagrangian or Lagrangean Relaxation skip

Giuseppe Ludovico Lagrangia (1736 Turin, 1813 Paris)
French ancestors on his father's side, he moved to Berlin and then Paris
Lagrangia (Joseph-Louis Lagrange) is alternatively considered a French and an Italian scientist

Lagrangean Relaxation

Given

$$\begin{aligned} \min\ & c^t x \\ & Ax \geq b \\ & x \in X \end{aligned}$$

Constraints are relaxed in the objective function

$$L(\pi) = \min_{x \in X} c^t x - \pi^t (Ax - b)$$

The lagrangean subproblem provides a LB:

$$L(\pi) \leq \min\{c^t x - \pi^t \underbrace{(Ax - b)}_{\geq 0} | Ax \geq b, x \in X\} \leq z$$

Lagrangean Relaxation

Given

$$\begin{aligned} \min\ & c^t x \\ & Ax \geq b \\ & x \in X \end{aligned}$$

Constraints are relaxed in the objective function

$$L(\pi) = \min_{x \in X} c^t x - \pi^t (Ax - b)$$

The lagrangean subproblem provides a LB:

$$L(\pi) \leq \min\{c^t x - \pi^t \underbrace{(Ax - b)}_{\geq 0} | Ax \geq b, x \in X\} \leq z$$

Lagrangean Relaxation

$$L(\pi) = \min_{x \in X} c^t x - \pi^t (Ax - b)$$

Lagrangean dual

$$\mathcal{L} = \max_{\pi \geq 0} L(\pi)$$

Given π, the Lagrange subproblem is

$$L(\pi) = \begin{cases} -\infty & \text{if } (c^t - \pi^t A)x_r < 0 \text{ for an } r \in R \\ c^t x_p - \pi^t (Ax_p - b) & \text{for some } p \in P \end{cases}$$

If we assume z to be finite, we rewrite the Lagrangean dual as

$$\max_{\pi \geq 0} \min_{p \in P} c^t x_p - \pi^t (Ax_p - b) \quad \texttt{s.t.} \quad (c^t - \pi^t A)x_r \geq 0 \forall r \in R$$

Lagrangean Relaxation

$$\max_{\pi \geq 0} \min_{p \in P} c^t x_p - \pi^t (A x_p - b) \quad \texttt{s.t.} \quad (c^t - \pi^t A) x_r \geq 0 \forall r \in R$$

Rewritten as a linear program

$$\begin{aligned} \mathcal{L} = \max \ & \pi_0 \\ \texttt{s.t.} \ & \pi^t (A x_p - b) + \pi_0 \leq c^t x_p, && p \in P \\ & \pi^t (A x_r) \leq c^t x_r, && r \in R \\ & \pi \geq 0 \end{aligned}$$

Lagrangean Relaxation

$$\begin{aligned} \mathcal{L} = \max\ & \pi_0 \\ \texttt{s.t.}\ & \pi^t(Ax_p - b) + \pi_0 \leq c^t x_p, && p \in P \\ & \pi^t(Ax_r) \leq c^t x_r, && r \in R \\ & \pi \geq 0 \end{aligned}$$

and the dual is ...

$$\begin{aligned} D(\mathcal{L}) = \min\ & \sum_{p \in P} c^t x_p \lambda_p + \sum_{r \in R} c^t x_r \lambda_r \\ \texttt{s.t.}\ & \sum_{p \in P} A x_p \lambda_p + \sum_{r \in R} A x_r \lambda_r \geq b \sum_{p \in P} \lambda_p \\ & \sum_{p \in P} \lambda_p = 1 \\ & \lambda \geq 0 \end{aligned}$$

Lagrangean Relaxation

For optimal x and π we obtain $z^* = c^t x = L(\pi)$
Lagrangean relaxation and Column generation form a dual pair
Remarks:

- DW decomposition satisfies complementary slackness conditions $(x \in conv(X), Ax \geq b)$
- Integrality must be checked for x
- Lagrangean relaxation: given π, x_π is integer feasible for X and $\pi^t(Ax_\pi - b) = 0$
- We have to check $Ax_\pi \geq b$

Dual interpretation of Column Generation

Column generation is a primal method

- primal (RMP) feasibility is always guaranteed
- unfeasible dual solution

Adding variables (columns) in the primal equals adding constraints in the dual.

Dual view

Each new variable (column of the primal) "cuts away" part of the infeasible dual region at every iteration

Dual interpretation of Column Generation

Column generation is a primal method

- primal (RMP) feasibility is always guaranteed
- unfeasible dual solution

Adding variables (columns) in the primal equals adding constraints in the dual.

Dual view

Each new variable (column of the primal) "cuts away" part of the infeasible dual region at every iteration

Dual interpretation of Column Generation

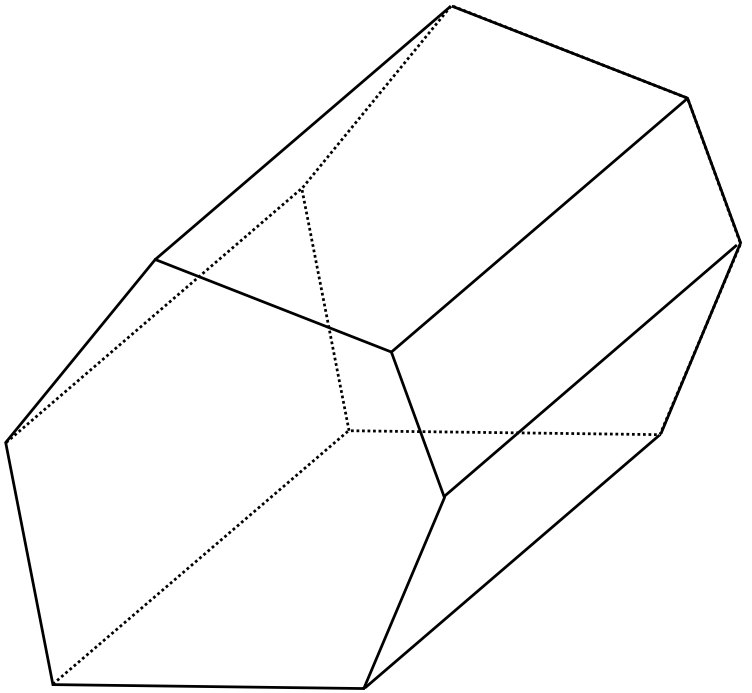
The development of dual variables toward their optima has been identified a major efficiency issue

In the beginning, generated variables are "irrelevant" because the initial dual information is meaningless (**heading-in**)

Conversely, quality of columns is best in the end, but they may be hard to generate, and proving optimality may take time (**tailing-off**)

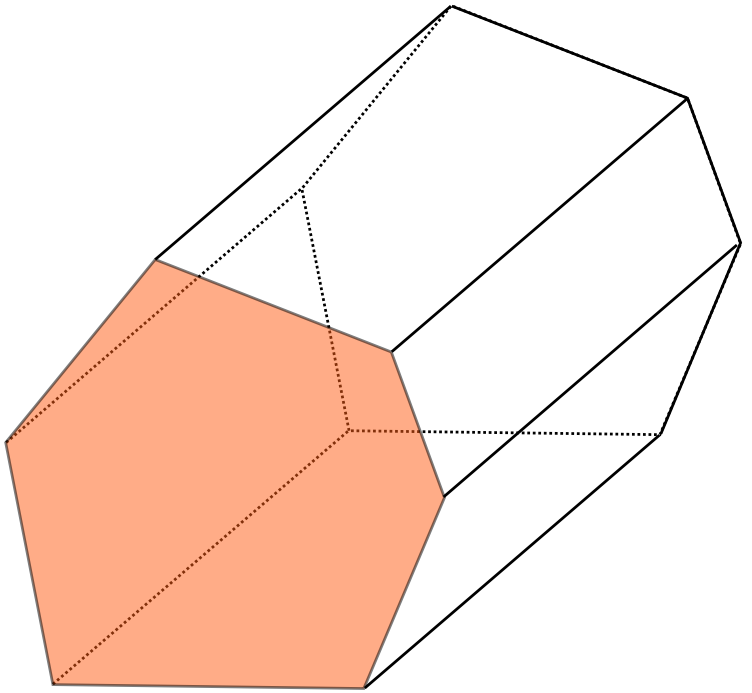
Degeneracy and Dual View

An RMP usually shows degeneracy: many variables are in the basis at 0 value.



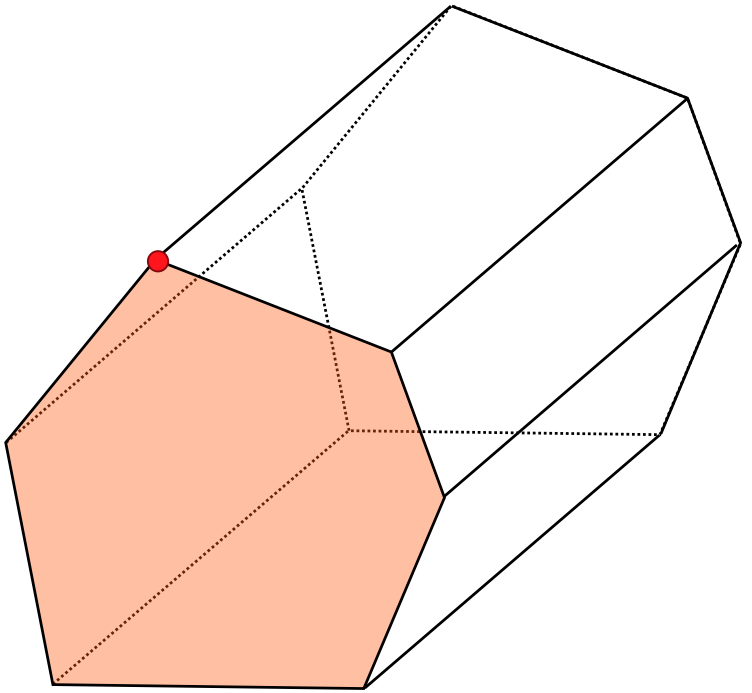
Degeneracy and Dual View

This means, in the dual space, that we have an optimal face rather than an optimal vertex.



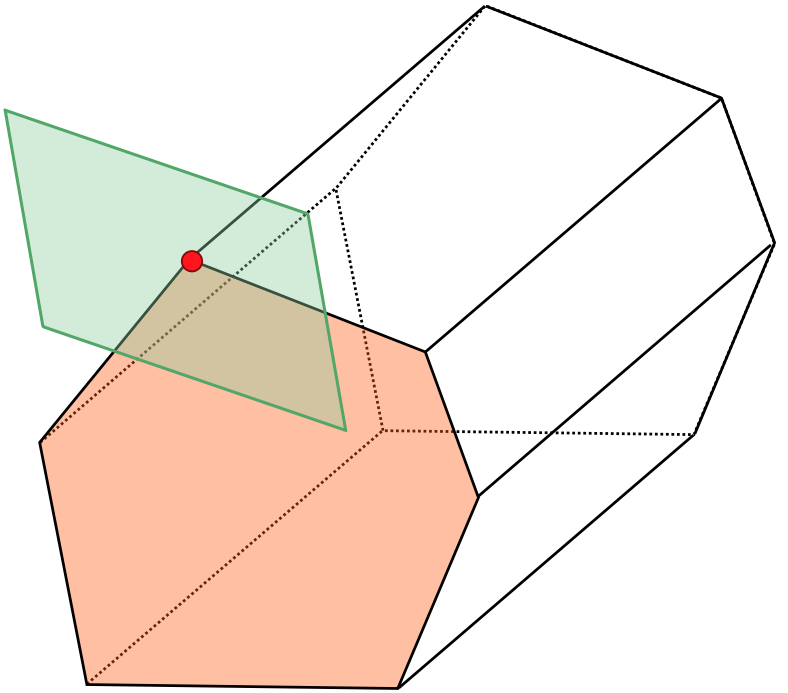
Degeneracy and Dual View

A simplex method converges to a **vertex** of the dual polyhedron.



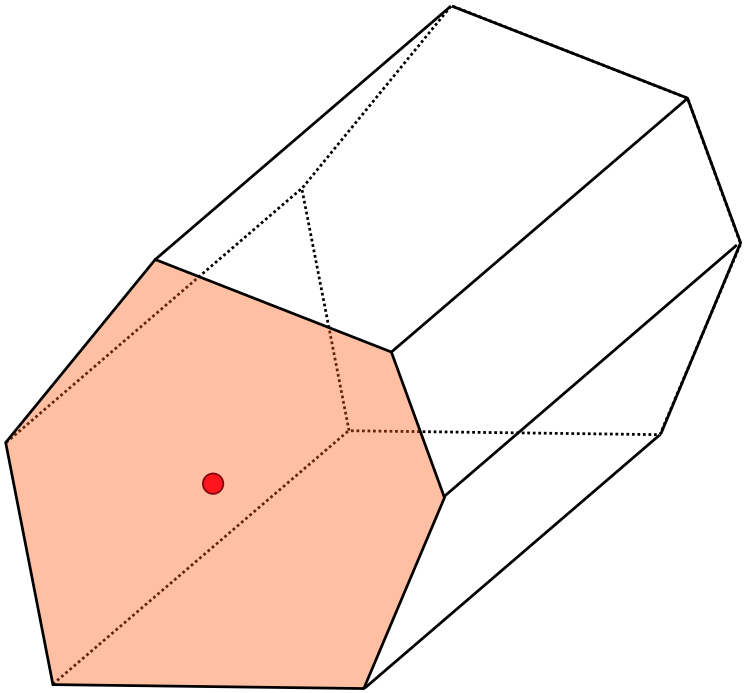
Degeneracy and Dual View

A column may cut away only that vertex or just a bit more



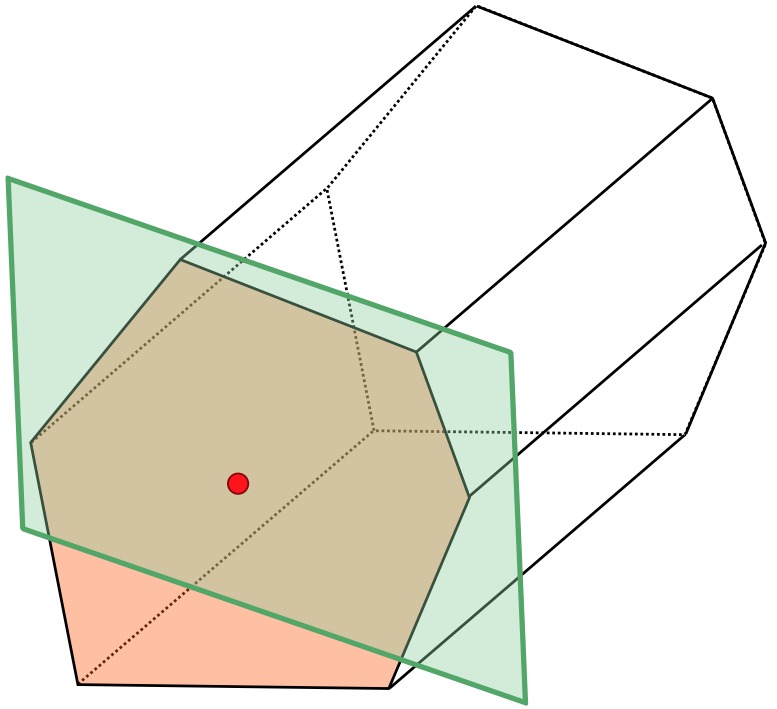
Degeneracy and Dual View

A “better” dual optimum



Degeneracy and Dual View

Could yeld a deeper cut



Dual space stabilization

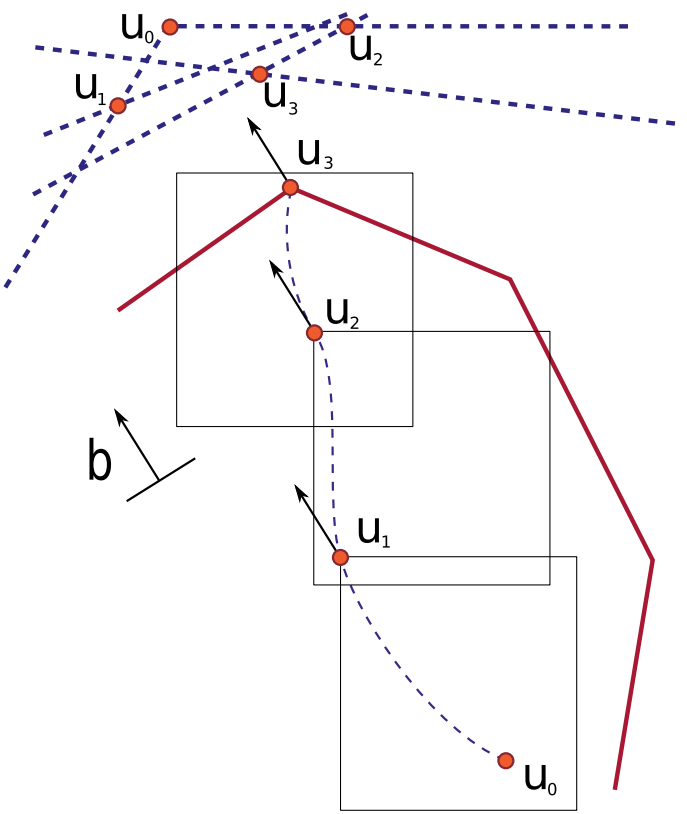
Linear program P

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned}$$

and its dual D

$$\begin{aligned} \max\ & b^T u \\ \text{s.t. } & A^T u \leq c \end{aligned}$$

Dual space stabilization - Box method



Dual space stabilization - Stabilized CG

Linear program $\tilde{P}$

$$\begin{aligned} \min\ & c^T x - \delta_- y_- + \delta_+ y_+ \\ \text{s.t. } & Ax - y_- + y_+ = b \\ & y_- \leq \varepsilon_- \\ & y_+ \leq \varepsilon_+ \\ & x, y_-, y_+ \geq 0 \end{aligned}$$

and its dual $\tilde{D}$

$$\begin{aligned} \max\ & b^T u - \varepsilon_-^T w_- - \varepsilon_+^T w_+ \\ \text{s.t. } & A^T u \leq c \\ & -u - w_- \leq -\delta_- \\ & u - w_+ \leq \delta_+ \\ & w_-, w_+ \geq 0 \end{aligned}$$

This equals to
$\delta_- - w_- \leq u \leq \delta_+ + w_+$

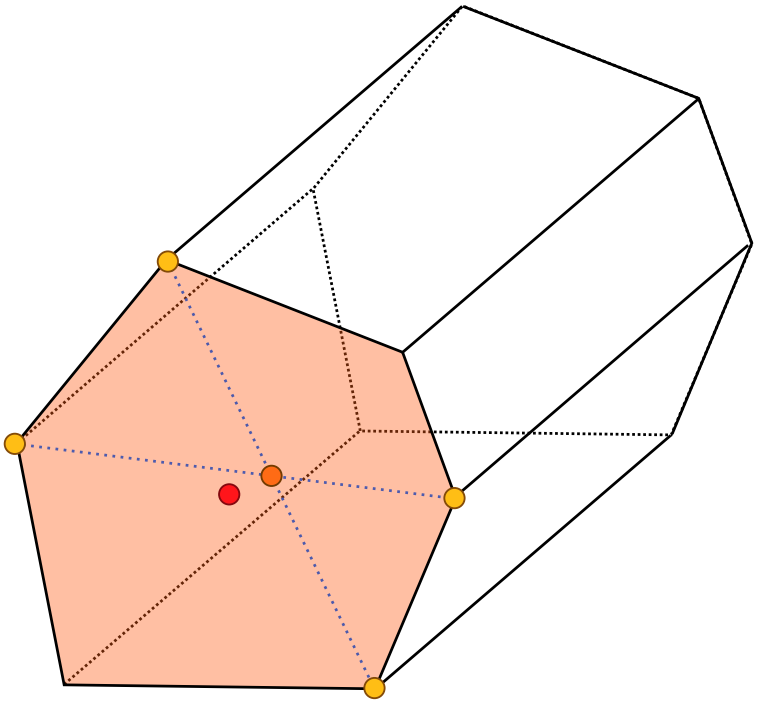
Dual space stabilization - Stabilized CG

Under what circumstances $\tilde{P}$ yield the optimal solution?

- $\varepsilon_- = \varepsilon_+ = 0$
- $\delta_- < u^* < \delta_+$

Dual space stabilization - Interior point method skip

Generate **several** vertices of the dual optimal face, obtain an interior point with a convex combination



Dual space stabilization
Interior point method

A set covering problems P

$$\min \sum_{r \in R} c_r x_r$$
$$\text{s.t.} \sum_{r \in R} a_{ir} x_r \geq 1 \quad \forall i \in \{1..N\}$$
$$x \geq 0$$

and its dual D

$$\max \sum_{i \in 1..N} u_i$$
$$\text{s.t.} \sum_{i \in 1..N} u_i a_{ir} \leq c_r \quad \forall r \in R$$
$$u \geq 0$$

Dual space stabilization
Interior point method

- Let $(\mathbf{x}, \mathbf{u})$ be the optimal primal and dual solution on P and D
- Let R^* be the set of columns for which $x_r > 0$
- Let S be the set of rows for which $\sum_{r \in R} a_{ir} x_r > 1$

Dual space stabilization
Interior point method

Using complementary slackness the dual polyhedron is:

$$\sum_{i \in 1..N} u_i a_{ir} \leq c_r \qquad \forall r \in R \setminus R^*$$

$$\sum_{i \in 1..N} u_i a_{ir} = c_r \qquad \forall r \in R^*$$

$$u_i = 0 \qquad \forall i \in S$$

$$u_i \geq 0 \qquad \forall i \in \{1..N\} \setminus S$$

Dual space stabilization
Interior point method

Define an optimization problem (D^{λ}), $\lambda_i \in U(0,1)$

$$\max \sum_{i \in 1..N} \lambda_i u_i$$

$$\sum_{i \in 1..N} u_i a_{ir} \leq c_r \qquad \forall r \in R \setminus R^*$$

$$\sum_{i \in 1..N} u_i a_{ir} = c_r \qquad \forall r \in R^*$$

$$u_i = 0 \qquad \forall i \in S$$

$$u_i \geq 0 \qquad \forall i \in \{1..N\} \setminus S$$

Use a **simplex based** method

Dual space stabilization
Interior point method

Define an optimization problem (D^{λ}), $\lambda_i \in U(0,1)$

$$\max \sum_{i \in 1..N} \lambda_i u_i$$

$$\sum_{i \in 1..N} u_i a_{ir} \leq c_r \qquad \forall r \in R \setminus R^*$$

$$\sum_{i \in 1..N} u_i a_{ir} = c_r \qquad \forall r \in R^*$$

$$u_i = 0 \qquad \forall i \in S$$

$$u_i \geq 0 \qquad \forall i \in \{1..N\} \setminus S$$

Use a **simplex based** method

Dual space stabilization
Interior point method

- Obtain several extreme points
- It is worth to solve (D^{λ}) and $(D^{-\lambda})$
- Take the average of all obtained extreme points
- We should write and manage explicitly the dual problem

Dual space stabilization
Interior point method

- Obtain several extreme points
- It is worth to solve (D^{λ}) and $(D^{-\lambda})$
- Take the average of all obtained extreme points
- We should write and manage explicitly the dual problem

Interior point method
In practice

Let's write the dual of (D^λ), (P^λ)

$$\max \sum_{i \in 1..N} \lambda_i u_i$$

$$\sum_{i \in 1..N} u_i a_{ir} \leq c_r \qquad \forall r \in R \setminus R^*$$

$$\sum_{i \in 1..N} u_i a_{ir} = c_r \qquad \forall r \in R^*$$

$$u = 0 \qquad \forall i \in S$$

$$u \geq 0 \qquad \forall i \in \{1..N\} \setminus S$$

$$\min \sum_{r \in R} c_r x_r$$

$$\sum_{r \in R} a_{ir} x_r \geq \lambda_i \qquad \forall i \in \{1..N\} \setminus S$$

$$\sum_{r \in R} a_{ir} x_r \geq -\infty \qquad \forall i \in S$$

$$x_r \geq 0 \qquad \forall r \in R \setminus R^*$$

$$x_r \ \textit{free} \qquad \forall r \in R^*$$

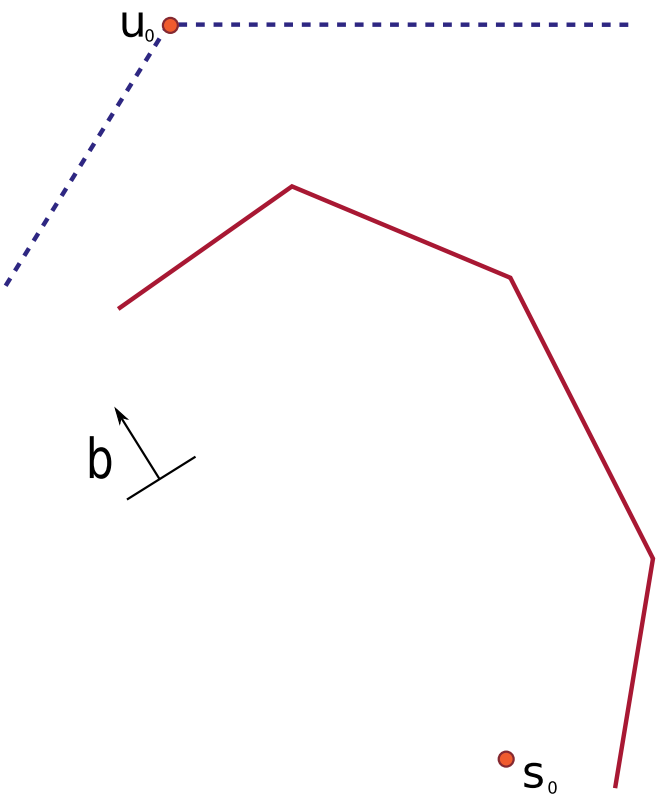
Interior point method
In practice

Compare (P) with (P^{λ})

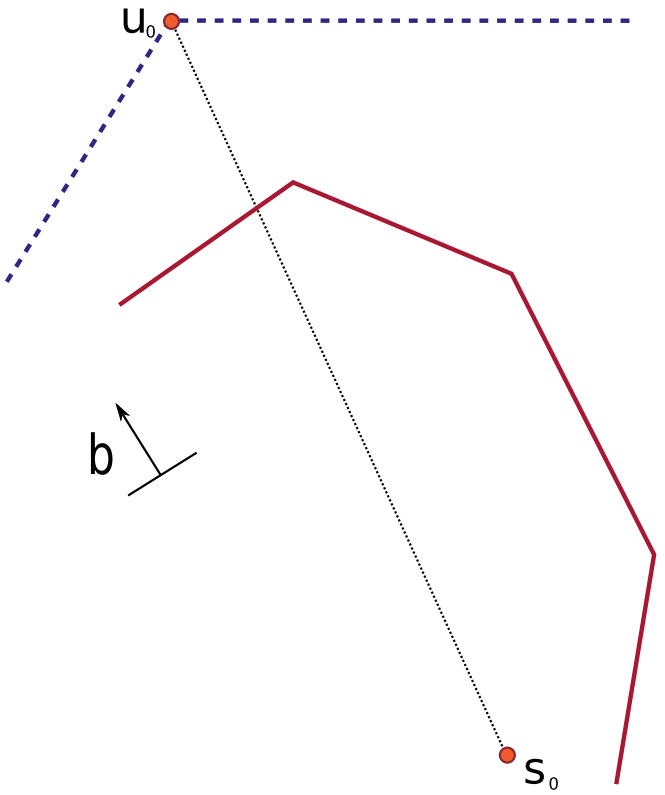
$$\begin{aligned}
\min \quad & \sum_{r \in R} c_r x_r & & \\
\text{s.t.} \quad & \sum_{r \in R} a_{ir} x_r \geq 1 & & \forall i \in \{1..N\} \\
& x \geq 0 & &
\end{aligned}$$

$$\begin{aligned}
\min \quad & \sum_{r \in R} c_r x_r & & \\
& \sum_{r \in R} a_{ir} x_r \geq \lambda_i & & \forall i \in \{1..N\} \setminus S \\
& \sum_{r \in R} a_{ir} x_r \geq -\infty & & \forall i \in S \\
& x_r \geq 0 & & \forall r \in R \setminus R^* \\
& x_r \ \textit{free} & & \forall r \in R^*
\end{aligned}$$

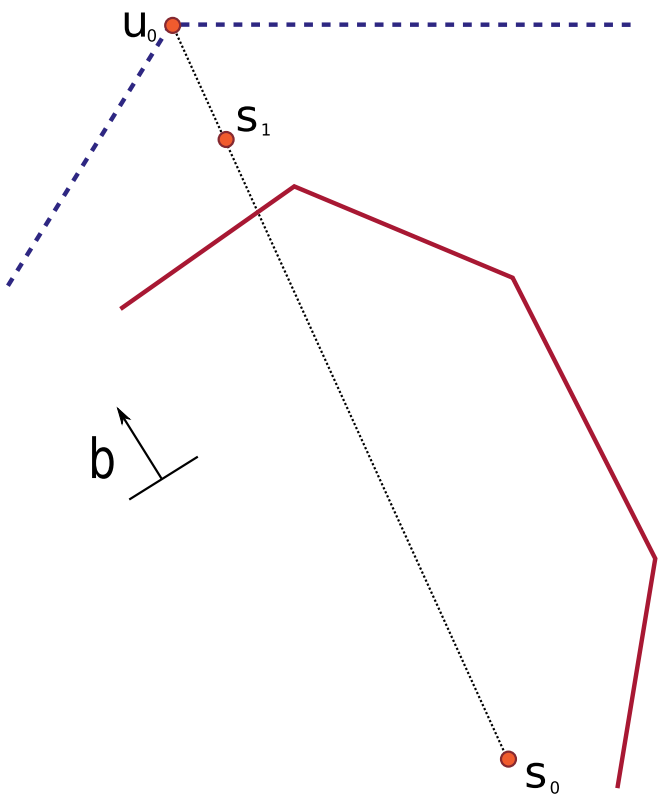
Dual space stabilization - Stability center method



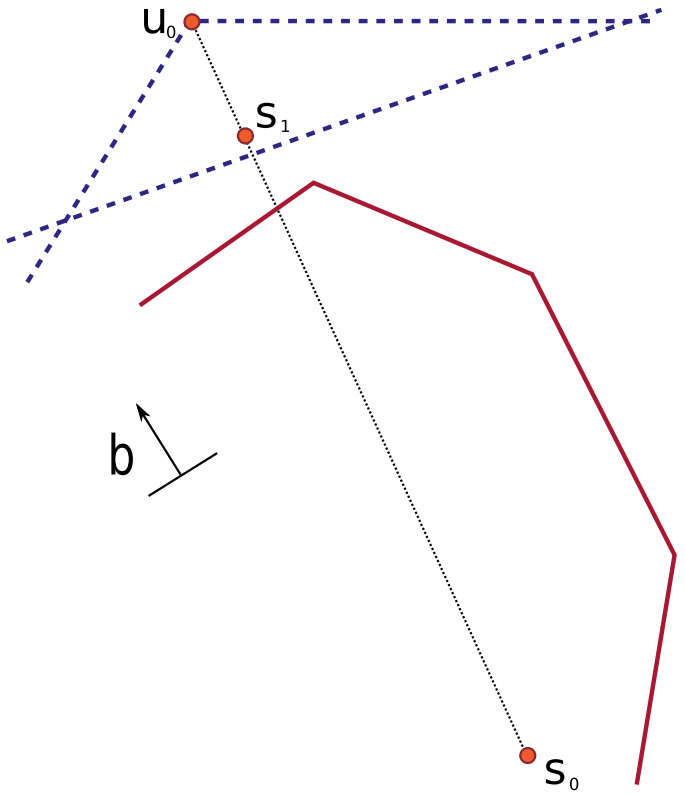
Dual space stabilization - Stability center method



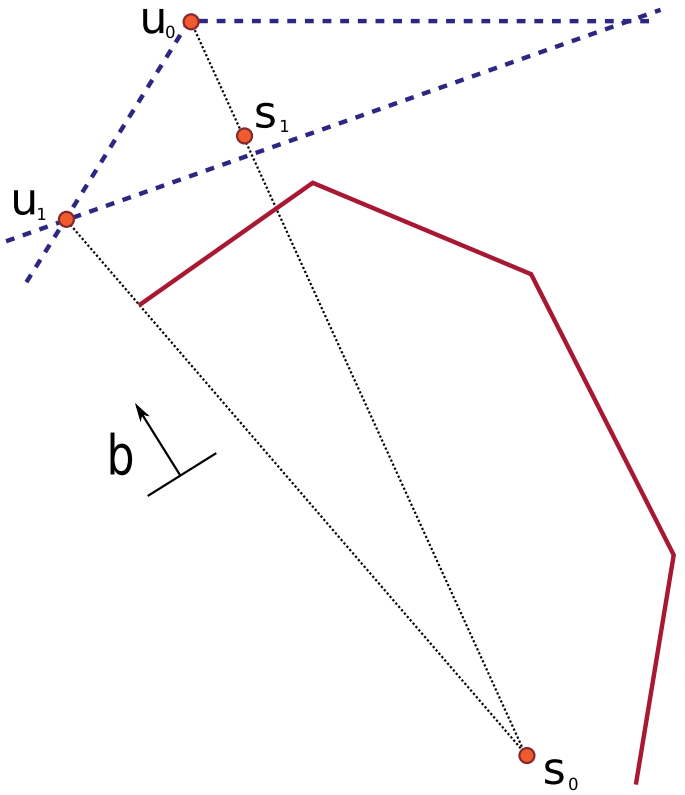
Dual space stabilization - Stability center method



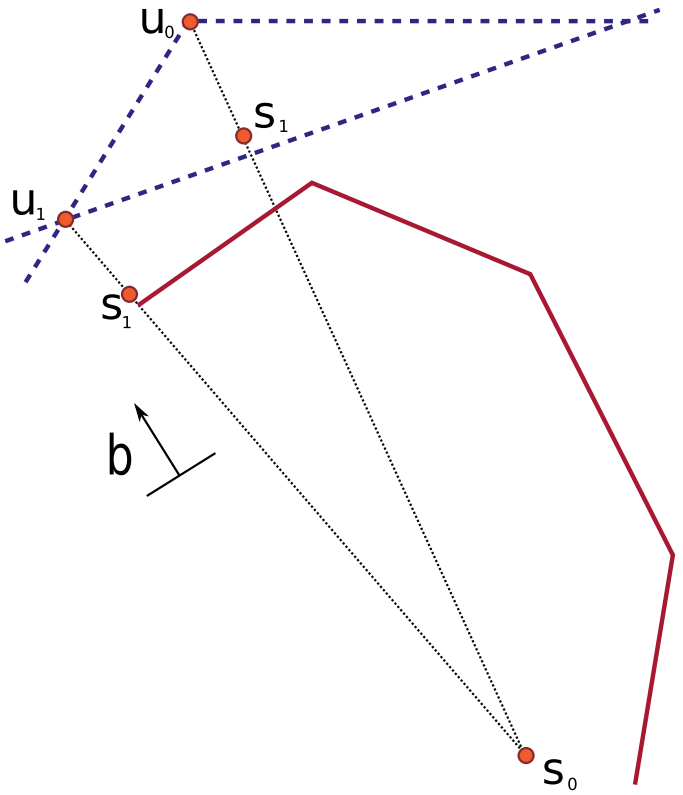
Dual space stabilization - Stability center method



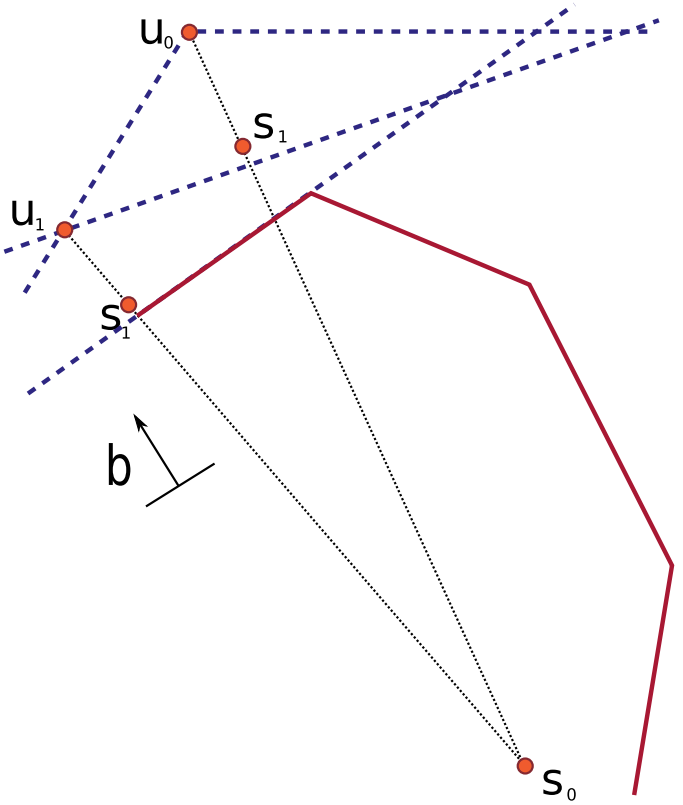
Dual space stabilization - Stability center method



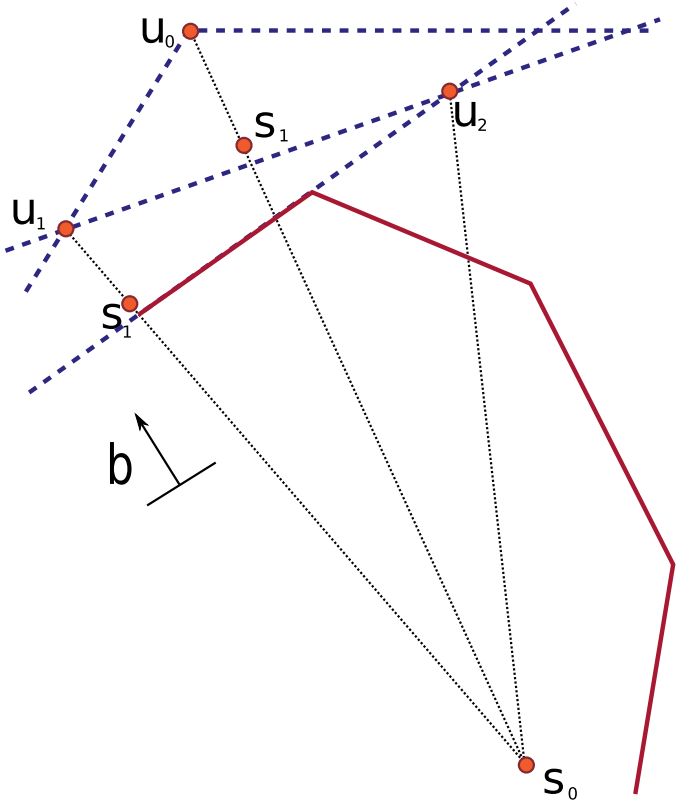
Dual space stabilization - Stability center method



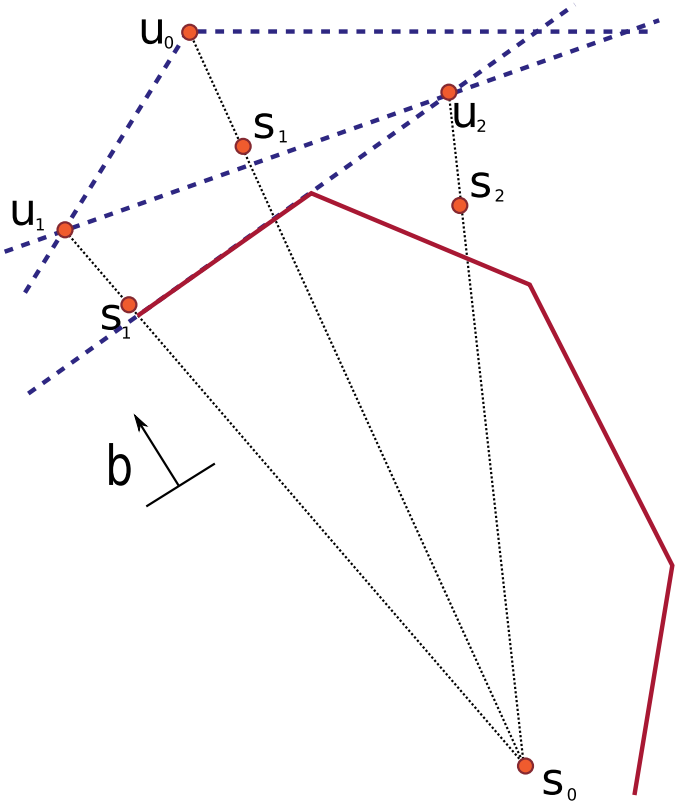
Dual space stabilization - Stability center method



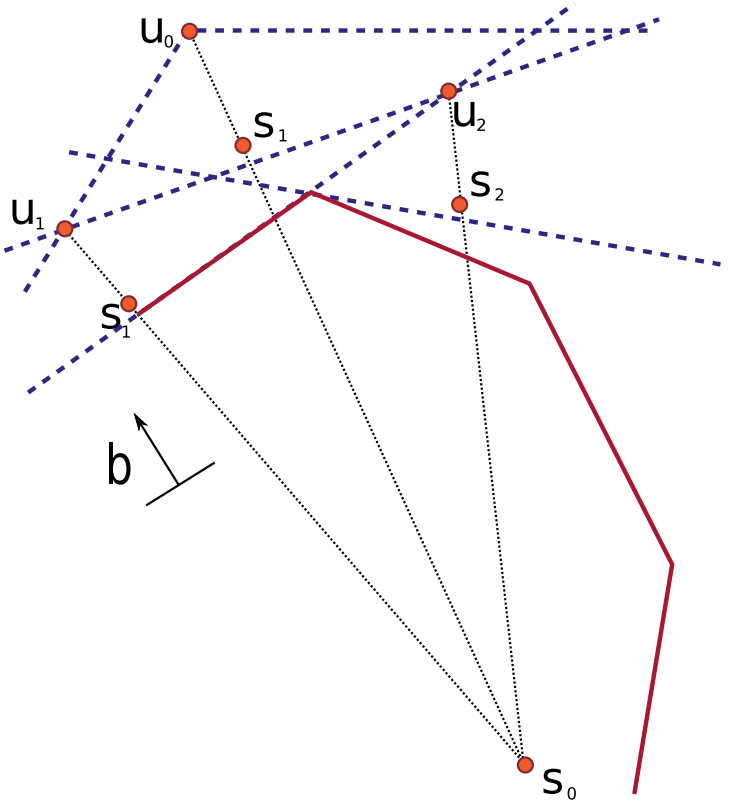
Dual space stabilization - Stability center method



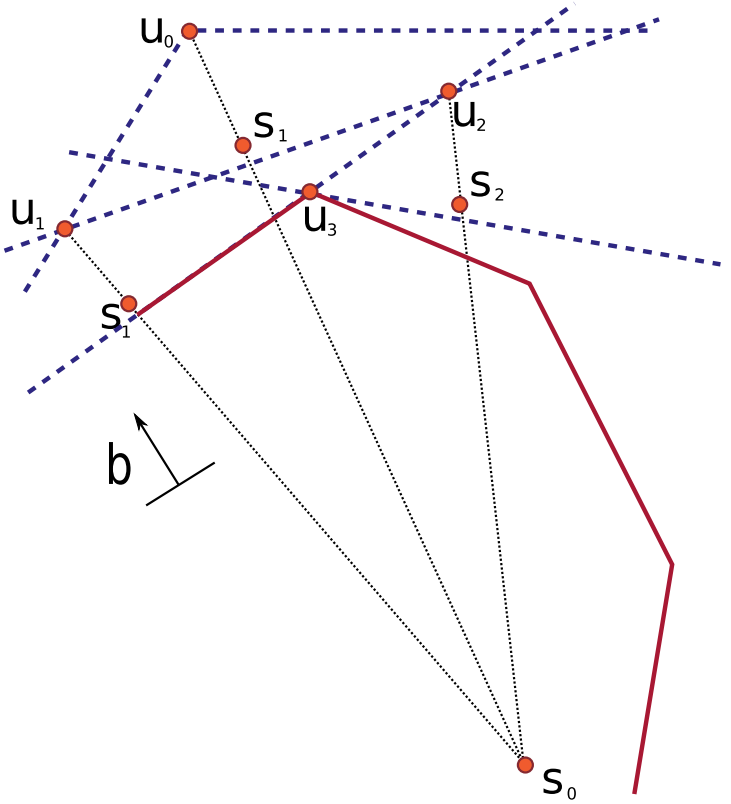
Dual space stabilization - Stability center method



Dual space stabilization - Stability center method



Dual space stabilization - Stability center method



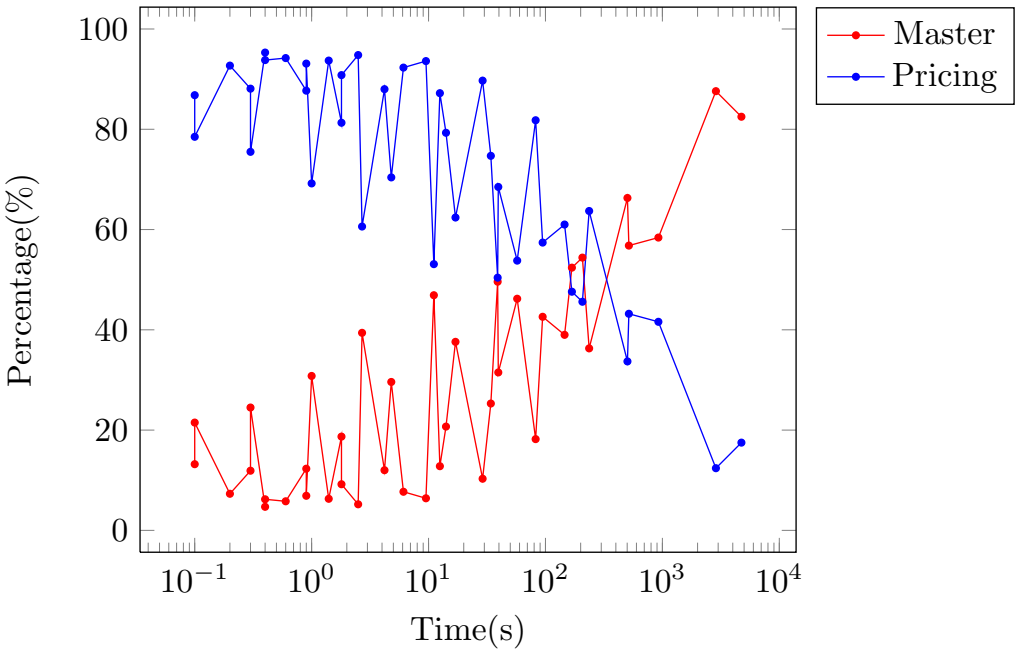
Dual space stabilization - Stability center method

Require: $\alpha, 0 < \alpha \leq 1$ and ε
  $\mathbf{s} = 0$
  repeat
    Solve RMP, obtain $(\underline{Z}_{EF}, \mathbf{u})$;
    $\mathbf{u}_{ST} = \alpha\mathbf{s} + (1-\alpha)\mathbf{u}$;
    Solve the pricing with $\mathbf{u}_{ST}$, obtain $\mathbf{a}_j$;
    if $L(\mathbf{u}_{ST}) > L(\mathbf{s})$ **then**
      Update $L(\mathbf{s})$ and $\mathbf{s}$;
    end if
    if $rc(\mathbf{a}_j) < 0$ **then**
      Add $\mathbf{a}_j$ to RMP;
    end if
  until $\underline{Z}_{EF} - L(\bar{\mathbf{u}}) \leq \varepsilon$

Stabilization - Practice

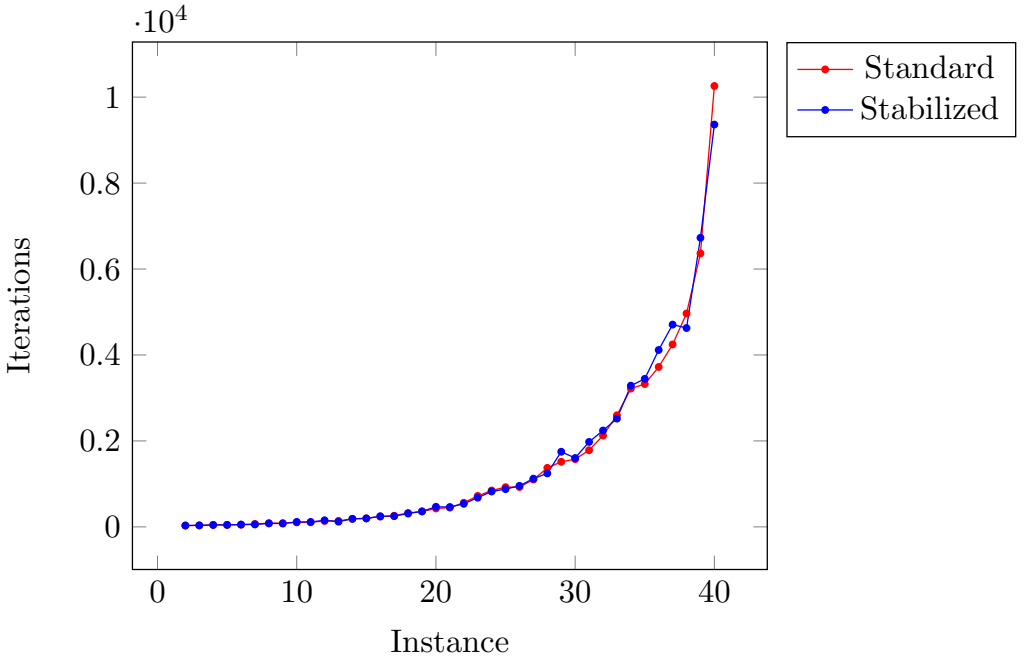
- For problems with high degeneracy (easy pricing problem, e.g., knapsack, SPP) stabilization is usually beneficial.
- Problems with very hard pricing (e.g., RCESPP) obtain less benefits from stabilization.

A good profiling practice suggests what to do:



Stabilization - Practice

(Personal implementation of) Stability center methods against interior point LP implemented in modern solvers.



Advanced topics in Column Generation

Some advanced topics related to CG:

- Reduced cost based variable elimination
- **Two-Stage Column Generation / state space reduction (mainly of pricing subproblems)**
- **Embedded relaxation**
- Dual optimal cuts
- Dynamic constraint aggregation
- Primal heuristics
- **Automatic decomposition**

Two-stage column generation

Compact formulation

$$
\begin{aligned}
z_{CF} = \min \quad & \mathbf{c^t x} \\
\text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b} \\
& D\mathbf{x} \geq \mathbf{d} \\
& \mathbf{x} \geq 0.
\end{aligned}
$$

Extensive formulation

$$
\begin{aligned}
z_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \lambda_p = 1 \\
& \lambda \geq 0, \ \mathbf{x} \geq 0.
\end{aligned}
$$

- Both the compact and the extensive formulations are managed dynamically starting from $\bar{X} \subset X$ and $\bar{P} \subset P$
- Pricing of the compact formulation variables is based on their reduced cost evaluation in the strengthened reformulation

Two-stage column generation

Require: set $\bar{X}$ ($\widehat{X} = X \setminus \bar{X}$)
  repeat {CG2: generate compact formulation variables $x \in \widehat{X}$}
    repeat {CG1: generate extensive variables λ}
      $\tilde{p} \leftarrow \operatorname{argmin}_{p \in \bar{P}} \tilde{c}_p$
      if $\tilde{c}_{\tilde{p}} < 0$ **then** $\bar{P} \leftarrow \bar{P} \cup \{\tilde{p}\}$
    until $\tilde{c}_{\tilde{p}} \geq 0$
    $\tilde{x} \leftarrow \operatorname{argmin}_{x \in \widehat{X}} \tilde{c}_{EF}(x)$
    if $\tilde{c}_{EF}(\tilde{x}) < 0$ **then** $\bar{X} \leftarrow \bar{X} \cup \{\tilde{x}\}$
  until $\tilde{c}_{EF}(\tilde{x}) \geq 0$

Extensive reduced cost

$$\tilde{c}_{EF}(x \in \widehat{X}) = \min_{p \in \{P | x > 0\}} \tilde{c}_p = \min\{c_p - \pi a_p - \pi_0 \mid p \in P \wedge x > 0\}.$$

Two-stage column generation

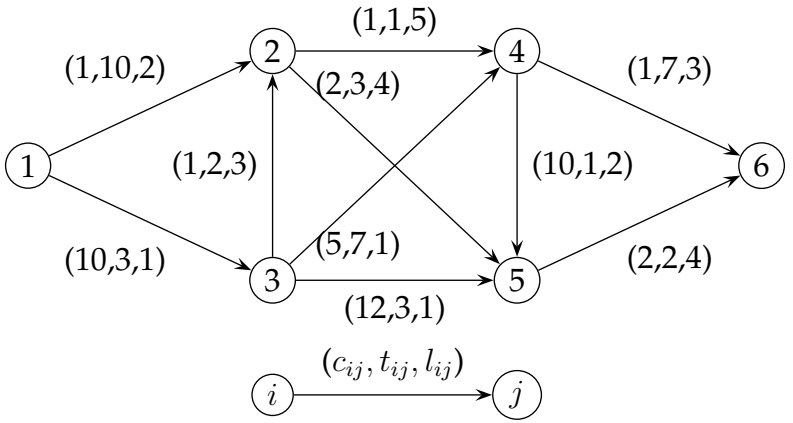
Require: set $\bar{X}$ ($\widehat{X} = X \setminus \bar{X}$)
  repeat {CG2: generate compact formulation variables $x \in \widehat{X}$}
    repeat {CG1: generate extensive variables λ}
      $\tilde{p} \leftarrow \operatorname{argmin}_{p \in \bar{P}} \tilde{c}_p$
      if $\tilde{c}_{\tilde{p}} < 0$ **then** $\bar{P} \leftarrow \bar{P} \cup \{\tilde{p}\}$
    until $\tilde{c}_{\tilde{p}} \geq 0$
    $\tilde{x} \leftarrow \operatorname{argmin}_{x \in \widehat{X}} \tilde{c}_{EF}(x)$
    if $\tilde{c}_{EF}(\tilde{x}) < 0$ **then** $\bar{X} \leftarrow \bar{X} \cup \{\tilde{x}\}$
  until $\tilde{c}_{EF}(\tilde{x}) \geq 0$

Extensive reduced cost

$$\tilde{c}_{EF}(x \in \widehat{X}) = \min_{p \in \{P | x > 0\}} \tilde{c}_p = \min\{c_p - \pi a_p - \pi_0 \mid p \in P \wedge x > 0\}.$$

Illustration of two-stage CG - RCESPP skip

Resource constrained shortest path



$$z_{CF} = \min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\sum_{j:(s,j)\in A} x_{sj} = 1$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \;\; \forall i \in A,\; i \neq s,t$$

$$\sum_{i:(i,t)\in A} x_{it} = 1$$

$$\sum_{(i,j)\in A} l_{ij} x_{ij} \leq L$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leq T$$

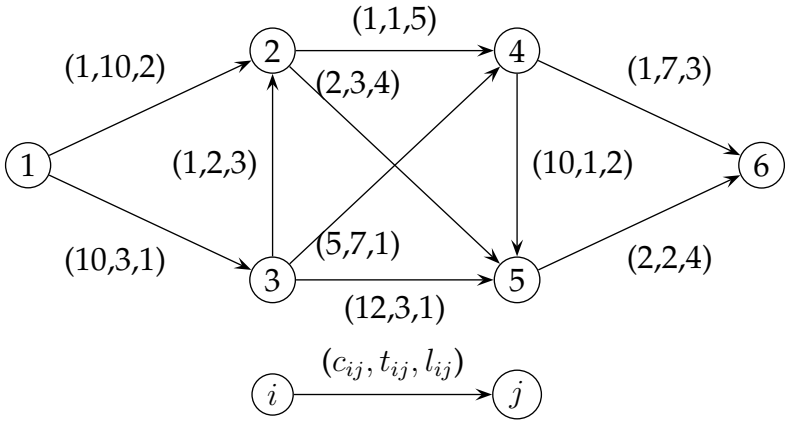
$$x_{ij} = \{0,1\} \;\; \forall (i,j) \in A$$

p	path	cost	time	load
1	1-2-4-6	3	18	10
2	1-2-5-6	5	15	10
3	1-2-4-5-6	14	14	13
4	1-3-5-6	24	8	6
5	1-3-4-6	16	17	5
6	1-3-4-5-6	27	13	6
7	1-3-2-4-6	13	13	12
8	1-3-2-4-5-6	24	9	15
9	1-3-2-5-6	15	10	12

$T = 14 \;\; L = 10$

Illustration of two-stage CG - RCESPP

Resource constrained shortest path



it	master obj	π_0	π_T	pricing obj	added path
1	100	100	0	-97.00	1-2-4-6
2	24.5	100	-5.39	-32.88	1-3-5-6
3	11.4	40.8	-2.10	-4.3	1-2-5-6
4	7.71	45.7	-2.71	0.0	STOP

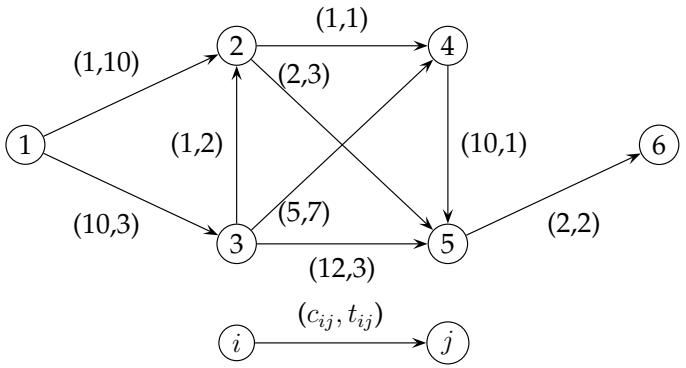
$z^*_{CF} = 7.2941$ $<$ $z^*_{EF} = 7.71$

$x^*_{12} = 0.82$, $x^*_{13} = 0.18$, $x^*_{25} = 0.94$, $\lambda^*_{1256} = 0.86$

$x^*_{32} = 0.12$, $x^*_{35} = 0.06$ and $x^*_{56} = 1$ $\lambda^*_{1356} = 0.14$

Illustration of two-stage CG - $\tilde{c}_{EF}$ method

$\widehat{X} = \{(4,6)\}$



it	master obj	added path
1.1	100.00	1-2-5-6
1.2	11.33	1-3-5-6
1.3	7.71	STOP

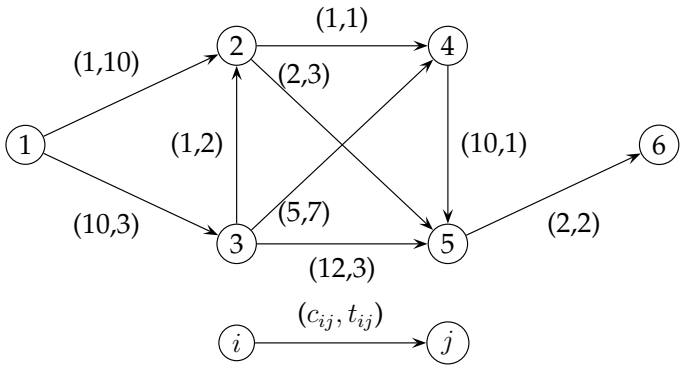
path	$\tilde{c}_p$	l_p	*note*
1-2-4-6	6.14	10	
1-2-5-6	0.00	10	
1-2-4-5-6	6.29	13	$l_p > L$
1-3-5-6	0.00	6	
1-3-4-6	16.43	5	
1-3-4-5-6	16.57	6	
1-3-2-4-6	2.57	12	$l_p > L$
1-3-2-4-5-6	2.71	15	$l_p > L$
1-3-2-5-6	-3.57	12	$l_p > L$

$$\tilde{c}_{EF}(x_{46}) = \min_{q \in \{Q | x_{46} > 0\}} = \min\{6.14, 16.43\} = 6.14.$$

Non optimal arcs are detected correctly

Illustration of two-stage CG - $\tilde{c}_{EF}$ method

$\widehat{X} = \{(4,6)\}$



it	master obj	added path
1.1	100.00	1-2-5-6
1.2	11.33	1-3-5-6
1.3	7.71	STOP

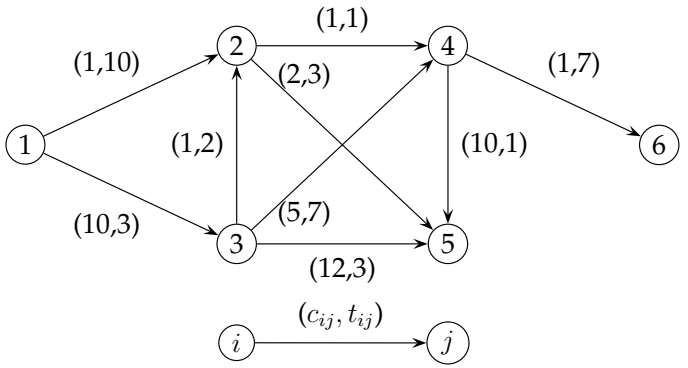
path	$\tilde{c}_p$	l_p	*note*
1-2-4-6	6.14	10	
1-2-5-6	0.00	10	
1-2-4-5-6	6.29	13	$l_p > L$
1-3-5-6	0.00	6	
1-3-4-6	16.43	5	
1-3-4-5-6	16.57	6	
1-3-2-4-6	2.57	12	$l_p > L$
1-3-2-4-5-6	2.71	15	$l_p > L$
1-3-2-5-6	-3.57	12	$l_p > L$

$$\tilde{c}_{EF}(x_{46}) = \min_{q \in \{Q | x_{46} > 0\}} = \min\{6.14, 16.43\} = 6.14.$$

Non optimal arcs are detected correctly

Illustration of two-stage CG - $\tilde{c}_{EF}$ method

$\widehat{X} = \{(5,6)\}$



it	master obj	added path
1.1	100.00	1-2-4-6
1.2	24.55	STOP
2.1	11.33	1-3-5-6
2.2	7.71	STOP

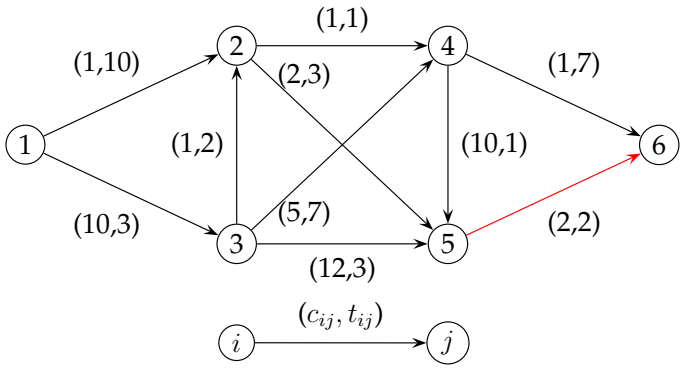
path	$\tilde{c}_p$	l_p	*note*
1-2-4-6	0.00	10	
1-2-5-6	-14.17	10	
1-2-4-5-6	-10.56	13	$l_p > L$
1-3-5-6	-32.89	6	
1-3-4-6	7.61	5	
1-3-4-5-6	-2.94	6	
1-3-2-4-6	-16.94	12	$l_p > L$
1-3-2-4-5-6	-27.50	15	$l_p > L$
1-3-2-5-6	-31.11	12	$l_p > L$

$$\tilde{c}_{EF}(x_{56}) = \min_{q \in \{Q | x_{56} > 0\}} = -32.89.$$

Optimal arcs are detected correctly

Illustration of two-stage CG - $\tilde{c}_{EF}$ method

$\widehat{X} = \emptyset$



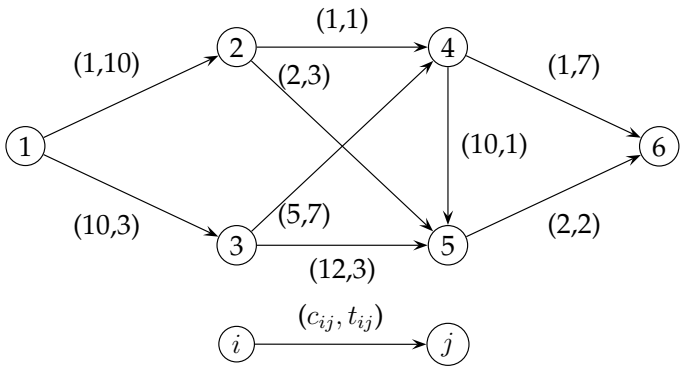
it	master obj	added path
1.1	100.00	1-2-4-6
1.2	24.55	STOP
2.1	11.33	1-3-5-6
2.2	7.71	STOP

path	$\tilde{c}_{EF}$	l_p	*note*
1-2-4-6	6.14	10	
1-2-5-6	0.00	10	
1-2-4-5-6	6.29	13	$l_p > L$
1-3-5-6	0.00	6	
1-3-4-6	16.43	5	
1-3-4-5-6	16.57	6	
1-3-2-4-6	2.57	12	$l_p > L$
1-3-2-4-5-6	2.71	15	$l_p > L$
1-3-2-5-6	-3.57	12	$l_p > L$

Optimal arcs are detected correctly

Illustration of two-stage CG - $\tilde{c}_{EF}$ method

$\widehat{X} = \{(3,2)\}$



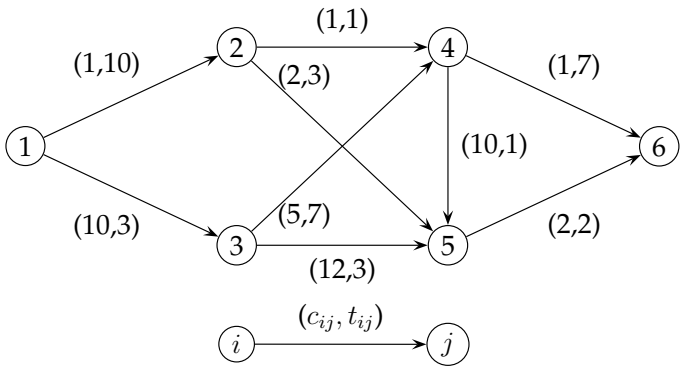
it	master obj	added path
1.1	100.00	1-2-4-6
1.2	24.55	1-3-5-6
1.3	11.40	1-2-5-6
1.4	7.71	STOP

path	$\tilde{c}_{EF}$	l_p	*note*
1-2-4-6	51.86	10	
1-2-5-6	45.71	10	
1-2-4-5-6	52.00	13	$l_p > L$
1-3-5-6	45.71	6	
1-3-4-6	62.14	5	
1-3-4-5-6	62.28	6	
1-3-2-4-6	48.28	12	$l_p > L$
1-3-2-4-5-6	48.43	15	$l_p > L$
1-3-2-5-6	42.14	12	$l_p > L$

$x_{3,2} \geq \varepsilon$, $\tilde{c}_{CF}(x_{32}) = -3.57143$

Illustration of two-stage CG - $\tilde{c}_{EF}$ method

$\widehat{X} = \{(3,2)\}$



it	master obj	added path
1.1	100.00	1-2-4-6
1.2	24.55	1-3-5-6
1.3	11.40	1-2-5-6
1.4	7.71	STOP

path	$\tilde{c}_{EF}$	l_p	*note*
1-2-4-6	51.86	10	
1-2-5-6	45.71	10	
1-2-4-5-6	52.00	13	$l_p > L$
1-3-5-6	45.71	6	
1-3-4-6	62.14	5	
1-3-4-5-6	62.28	6	
1-3-2-4-6	48.28	12	$l_p > L$
1-3-2-4-5-6	48.43	15	$l_p > L$
1-3-2-5-6	42.14	12	$l_p > L$

$x_{3,2} \geq \varepsilon$, $\tilde{c}_{CF}(x_{32}) = -3.57143$

Two stage CG : Application to DSDVRPTW

- Solomon's data set C and R instances;
- 25 and 50 customers;
- 3 scenarios: A,B, and C (3,5, and 7 orders/customer);
- up to 350 orders in total
- focus: **proving optimality of EF**
- time limit: $10h$

R_25_C - 175 orders

	Stand.CG		Two-stage CG		
Instance	**cols**	**t**	**cols**	**it**	**t**
R101_25_C_100	635	1	478	4	0
R102_25_C_100	2243	4	1159	4	6
R103_25_C_100	2848	11	2134	4	46
R104_25_C_100	3558	28	1198	5	79
R105_25_C_100	1668	4	854	4	2
R106_25_C_100	2151	6	1100	5	15
R107_25_C_100	2226	24	1355	4	41
R108_25_C_100	2807	27	1496	4	54
R109_25_C_100	2172	22	1334	4	7
R110_25_C_100	2313	20	948	4	45
R111_25_C_100	3081	33	1817	4	63
R112_25_C_100	3028	28	781	4	28
AVG	**2394**	**17**	**1221**	**4.2**	**32**

C_25_C - 175 orders

	Stand.CG		Two-stage CG		
Instance	**cols**	**t**	**cols**	**it**	**t**
C101_25_C_100	1498	24	854	4	18
C102_25_C_100	2907	332	1173	4	252
C103_25_C_100	3195	592	1598	4	633
C104_25_C_100	2381	27660	1704	4	33527
C105_25_C_100	1602	86	1043	4	113
C106_25_C_100	1697	34	1050	5	58
C107_25_C_100	1598	85	1070	4	355
C108_25_C_100	1981	187	1235	4	191
C109_25_C_100	1652	983	1039	4	3084
AVG	**2057**	**3331 (290)**	**1196**	**4.1**	**4248 (588)**

C_50_A - 150 orders

	Stand.CG		Two-stage CG		
Instance	**cols**	**t**	**cols**	**it**	**t**
C101_50_A_100	1204	9	898	4	4
C102_50_A_100	2022	344	1563	4	28
C103_50_A_100	1656	6844	1216	4	160
C105_50_A_100	1076	12	759	5	5
C106_50_A_100	1115	24	850	5	6
C107_50_A_100	1298	34	1222	5	17
C108_50_A_100	972	173	793	4	30
C109_50_A_100	1194	484	907	4	112
AVG	**1317**	**991**	**1026**	**4.4**	**45**

C_50_B - 250 orders

	Stand.CG		Two-stage CG		
Instance	**cols**	**t**	**cols**	**it**	**t**
C101_50_B_100	2407	77	1482	5	21
C102_50_B_100	2664	2555	1939	5	575
C103_50_B_100	x	>10h	2429	6	3519
C105_50_B_100	2660	323	1862	5	215
C106_50_B_100	2721	883	1808	5	63
C107_50_B_100	2535	865	1677	5	311
C108_50_B_100	2785	2951	2090	6	1044
C109_50_B_100	2849	5563	1764	7	4567
AVG	**2660**	**6152**	**1881**	**5.5**	**1289**

C_50_C - 350 orders

	Stand.CG		Two-stage CG		
Instance	**cols**	**t**	**cols**	**it**	**t**
C101_50_C_100	4757	589	2706	8	482
C105_50_C_100	4727	27736	2103	8	1654
C106_50_C_100	3864	1372	2127	7	551
C107_50_C_100	3958	1376	2185	7	4340
C108_50_C_100	4668	11274	2167	8	12413
AVG	**4395**	**8470**	**2258**	**7.6**	**3888**

Embedded relaxation

For many applications and/or during the development of the search tree, formulation may become harder:

$$
\begin{aligned}
z_{EF} = \min \quad & \sum_{p \in P} c_p \lambda_p \\
\text{s.t.} \quad & \sum_{p \in P} \mathbf{a_p} \lambda_p \geq \mathbf{b} \\
& \sum_{p \in P} \mathbf{f_p} \lambda_p \leq \mathbf{g} \qquad (1) \\
& \sum_{p \in P} \lambda_p = 1 \\
& \lambda \geq 0, \ \mathbf{x} \geq 0.
\end{aligned}
$$

Linear relaxation may satisfy the constraint (1) but no integer solution does.

Embedded relaxation

Simply the dynamic management of objective and constraints.

Require: $\tilde{P}$
  $Tree_P$:= Initialize root node; $z_P^* := +\infty$; $LB_{z_P} := 0$;
  while $Tree_P \neq \emptyset$ **do**
    $Node_P$:= extract node($Tree_P$);
    $EmbTree_Q$:= Initialize root node ($Node_P$); $z_Q^* := +\infty$; $LB_{z_Q} := 0$;
    Adjust coefficients in RMP from c_p to f_p, relax constraint (1);
    while $EmbTree_Q \neq \emptyset$ **do**
      $EmbTreeNode_Q$:= extract node($EmbTree_Q$);
      LB_{z_Q} := Solve Column Generation (f_p, $EmbTreeNode_Q$);
      if Integral(RMP) **and** $LB_{z_Q} < z_Q^*$ **then**
        $z_Q^* := LB_{z_Q}$;
      else if $LB_{z_Q} < z_Q^*$ **then**
        Branch($EmbTree_Q$);
      else
        Prune($EmbTreeNode_Q$);
      end if
    end while
    if $z_Q^* \leq$ **g then**
      LB_{z_P} := Solve Column Generation (c_p, $Node_P$);
      if Integral(RMP) **and** $LB_{z_P} < g_P^*$ **then**
        $z_P^* := LB_{z_P}$;
      else if $LB_{z_P} < z_P^*$ **then**
        Branch($Tree_P$);
      else
        Prune($Node_P$);
      end if
    else
      Prune($Node_P$);
    end if
  end while

Embedded relaxation

The key element of embedded relaxation is that the RMP is **"shared"** between formulations as the convexified polyhedron is the same: all columns are valid in both formulations. Furthermore all master cuts are valid in both formulations.

Minimal violation for VRP with Soft Time Windows

Instance	z^*_Ω	Improvement 1% g^*_Θ	Improvement 1% $T_{g^*}(s)$	Improvement 1% $T^E_{g^*}(s)$	Improvement 5% g^*_Θ	Improvement 5% $T^E_{g^*}(s)$	Improvement 10% g^*_Θ	Improvement 10% $T^E_{g^*}(s)$
r101_200	617.10	7.00	3.38	2.67	21.80	5.91	41.40	2.56
r102_200	547.10	2.40	65.12	38.25	2.40	68.23	29.40	1648.68
r103_200	454.60	7.00	-	16756.00	23.60	35905.99	*	0.22
r104_200	416.90	6.70	-	-	43.30	-	*	6.47
r105_200	530.50	2.20	23.65	12.20	19.60	30.03	48.90	41.98
r106_200	465.40	11.00	-	17185.00	21.00	11821.03	57.10	16205.24
r107_200	424.30	4.60	-	-	47.30	-	*	0.92
r108_200	397.30	2.30	-	-	27.60	-	*	2.4
r109_200	441.30	20.90	-	3079.91	*	2.09	*	2.11
r110_200	444.10	0.10	-	15964.30	10.20	22045.98	*	8.67
r111_200	428.80	3.90	-	28277.10	22.30	-	*	3.21
r112_200	393.00	21.00	-	-	59.00	-	*	0.89
r101_100	617.10	7.00	3.11	2.37	21.80	5.47	41.40	3.08
r102_100	547.10	2.40	87.14	19.94	2.40	48.87	29.40	1325.86
r103_100	454.60	7.00	-	11908.91	23.60	19055.03	*	0.2
r104_100	416.90	6.70	-	-	*	0.69	*	0.69
r105_100	530.50	2.20	28.88	10.8	19.60	30.4	48.90	41.18
r106_100	465.40	11.00	-	12798.32	25.80	7356.04	57.10	14056.04
r107_100	428.40	4.20	-	-	*	0.67	*	0.75
r108_100	403.20	2.30	-	-	*	1.79	*	1.88
r109_100	441.30	20.90	-	1408.5	*	0.86	*	0.89
r110_100	444.10	0.10	-	10415.33	10.20	11185.59	*	1.33
r111_100	428.80	3.90	-	26795.29	35.50	-	*	0.68
r112_100	401.70	*	-	95.99	*	97.31	*	8.66

Table : R Instances. Optimal results for objective improvement of 1%, 5% and 10%.

Automatic decomposition

Main purpose: automatize Dantzig-Wolfe reformulation without any user input or expert knowledge.
Briefly, given an integer program:

- A standard solver does not "see" any embedded structure.
- OR practitioners:
  - Are not aware of decomposition mechanisms and rely on existing solvers (regularly formulate models but cannot solve even moderately sized instances);
  - They know about decomposition but have no clue on how to practically use it;
  - They have an implementation of branch-and-price but don't know whether it is worth to modify it to address the new problem.
- Existing frameworks need the knowledge on $D\mathbf{x} \geq \mathbf{d}$.

Automatic decomposition

Key concept: automatic identification of structure in coefficient matrix.

$$D = \begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^k \end{pmatrix} \qquad D = \begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^k \\ A^1 & A^2 & \dots & A^k \end{pmatrix}$$

Given A, construct a hypergraph $H = (V, RC)$. For every $a_{ij} \neq 0$ associate a vertex $v_{ij} \in V$. Hyperedges $r_i \in R$ for $v_{r_i,j} \in V$ corresponding to non-zero entries of row i. The same for columns C. A is block diagonal if H decomposes in connected components.
Remove the minimal number of hyperedges from H to obtain a block-diagonal A.

Conclusions

- Decomposition principles are the step stones to attack large scale optimization problems.
- Dantzig-Wolfe decomposition is one of such (with Lagrangean and Benders').
- We should be able to solve the subproblem much more efficiently than the corresponding MILP.
- The stronger the bound the better it is: stronger formulations and additional cuts.
- Coordination between heuristics and relaxations.

Thanks!

The starting point for your personal exploration of CG can be:

Marco E. Lübbecke, 2011. *Column Generation*, Wiley Encyclopedia of Operations Research and Management Science. doi:10.1002/9780470400531.eorms0158

Jacques Desrosiers, Marco E. Lübbecke, 2011. *Branch-Price-and-Cut Algorithms*, Wiley Encyclopedia of Operations Research and Management Science. doi:10.1002/9780470400531.eorms0118