Branch-and-Price Crash Course

Marco Lübbecke

Chair of Operations Research · RWTH Aachen University

CO@Work 2024 · ZIB@40 Berlin · September 19, 2024

image by Darius Dan on `flaticon.com`

@mluebbecke@mas.to

Operations Research | RWTH AACHEN UNIVERSITY

Branch-and-Price is about

- extending your modeling capabilities
- algorithmically exploiting subproblems that you can solve well

The Cutting Stock Problem

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

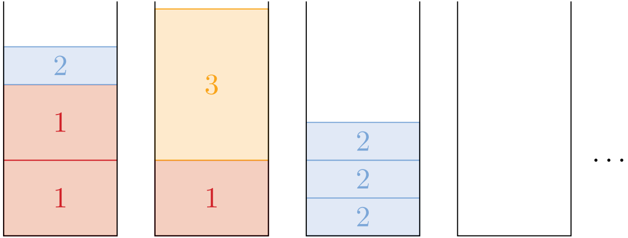
Example: The Cutting Stock Problem

Data

m rolls of raw material, each of length $W \geq 0$; and
n items of length $w_i \geq 0$ with a demand of $b_i \in \mathbb{Z}_+$, $i = 1, \ldots, n$

Goal

cut rolls into items, satisfying all demands, minimizing the number of used rolls



The Cutting Stock Problem: An “Assignment Type” Model

- formulation as an integer program; notation $[n] := \{1, \ldots, n\}$

The Cutting Stock Problem: An "Assignment Type" Model

- formulation as an integer program; notation $[n] := \{1, \ldots, n\}$

$$x_{ij} \in \mathbb{Z}_+ \qquad i \in [n], j \in [m] \quad \text{// how often to cut } i \text{ from } j$$

in our context: this is the "original" or "compact" formulation

The Cutting Stock Problem: An "Assignment Type" Model

- formulation as an integer program; notation $[n] := \{1, \dots, n\}$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} \geq b_i \qquad i \in [n] \qquad \text{// cover all demands}$$

$$x_{ij} \in \mathbb{Z}_+ \qquad i \in [n],\ j \in [m] \qquad \text{// how often to cut } i \text{ from } j$$

in our context: this is the "original" or "compact" formulation

The Cutting Stock Problem: An "Assignment Type" Model

- formulation as an integer program; notation $[n] := \{1, \ldots, n\}$

$$
\begin{array}{llll}
\text{s.t.} & \displaystyle\sum_{j=1}^{m} x_{ij} \geq b_i & i \in [n] & \text{// cover all demands} \\
& \displaystyle\sum_{i=1}^{n} w_i x_{ij} \leq W & j \in [m] & \text{// respect roll capacities} \\
& x_{ij} \in \mathbb{Z}_+ & i \in [n],\ j \in [m] & \text{// how often to cut } i \text{ from } j
\end{array}
$$

in our context: this is the "original" or "compact" formulation

The Cutting Stock Problem: An "Assignment Type" Model

- formulation as an integer program; notation $[n] := \{1, \ldots, n\}$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} \geq b_i && i \in [n] && \text{// cover all demands} \\
& \sum_{i=1}^{n} w_i x_{ij} \leq W y_j && j \in [m] && \text{// respect roll capacities} \\
& x_{ij} \in \mathbb{Z}_+ && i \in [n],\, j \in [m] && \text{// how often to cut } i \text{ from } j \\
& y_j \in \{0, 1\} && j \in [m] && \text{// do we use roll } j\text{?}
\end{aligned}
$$

in our context: this is the "original" or "compact" formulation

The Cutting Stock Problem: An "Assignment Type" Model

- formulation as an integer program; notation $[n] := \{1, \ldots, n\}$

$$
\begin{array}{llll}
\min & \displaystyle\sum_{j=1}^{m} y_j & & \text{// minimize number of used rolls} \\
\text{s.t.} & \displaystyle\sum_{j=1}^{m} x_{ij} \geq b_i & i \in [n] & \text{// cover all demands} \\
& \displaystyle\sum_{i=1}^{n} w_i x_{ij} \leq W y_j & j \in [m] & \text{// respect roll capacities} \\
& x_{ij} \in \mathbb{Z}_+ & i \in [n], j \in [m] & \text{// how often to cut } i \text{ from } j \\
& y_j \in \{0, 1\} & j \in [m] & \text{// do we use roll } j\text{?}
\end{array}
$$

in our context: this is the "original" or "compact" formulation

Observations on the Compact Model

i.e., they don't share any variables

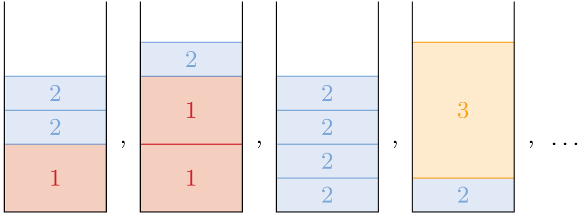
- the model contains m *independent* knapsack constraints ("local")
  // we know well how to solve knapsack problems

$$\sum_{i=1}^{n} w_i x_{ij} \leq W \quad j \in [m]$$ // respect roll capacities

- these are "coordinated" by the demand constraints ("global")
- the model is *symmetric* in index j: given a feasible solution, any permutation of the j indices gives an equivalent feasible solution // this is bad

Cutting Stock Problem: Kantorovich (1939), Gilmore & Gomory (1961)

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

$\mathcal{P} = \Bigg\{$  $\Bigg\}$

$$= \left\{ \begin{pmatrix}1\\2\\0\end{pmatrix}, \begin{pmatrix}2\\1\\0\end{pmatrix}, \begin{pmatrix}0\\4\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \ldots \right\}$$

$a_{1p} = 0$
$a_{2p} = 1$
$a_{3p} = 1$

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

$$\mathcal{P} = \left\{ \begin{pmatrix}a_1\\ \vdots \\ a_n\end{pmatrix} \in \mathbb{Z}_+^n \;\middle|\; \sum_{i=1}^{n} w_i a_i \leq W \right\}$$

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

Cutting Stock Problem: Kantorovich (1939), Gilmore & Gomory (1961)

- we build a model on these observations, on *entire configurations*

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

in our context: we call this an "extended" formulation // well, in fact, it *is* one

Cutting Stock Problem: Kantorovich (1939), Gilmore & Gomory (1961)

- we build a model on these observations, on *entire configurations*

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}} a_{ip} \lambda_p \geq b_i \qquad i \in [n] \qquad \text{// cover all demands}$$

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

in our context: we call this an "extended" formulation // well, in fact, it *is* one

Cutting Stock Problem: Kantorovich (1939), Gilmore & Gomory (1961)

- we build a model on these observations, on *entire configurations*

$$\min \sum_{p\in\mathcal{P}} \lambda_p \qquad \text{// minimimize number of patterns used}$$

$$\text{s.t.} \sum_{p\in\mathcal{P}} a_{ip}\lambda_p \geq b_i \qquad i \in [n] \qquad \text{// cover all demands}$$

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

in our context: we call this an "extended" formulation // well, in fact, it *is* one

- how can we solve even only the LP relaxation of such models?

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation

3 Branch-Price-and-Cut

Column Generation to solve a Linear Program

- we want to solve the *master problem* (MP)

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

- typically, $|J|$ huge

Column Generation to solve a Linear Program

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

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

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

Column Generation to solve a Linear Program

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

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

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

Column Generation to solve a Linear Program

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

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

- typically, $|J|$ huge, $|J'|$ small
- use e.g., simplex method to obtain optimal primal $\boldsymbol{\lambda}$ and optimal dual $\boldsymbol{\pi}$
- does $\boldsymbol{\lambda}$ solve the master problem to optimality as well?

Column Generation to solve a Linear Program

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

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

- typically, $|J|$ huge, $|J'|$ small
- use e.g., simplex method to obtain optimal primal $\boldsymbol{\lambda}$ and optimal dual $\boldsymbol{\pi}$
- does $\boldsymbol{\lambda}$ solve the master problem to optimality as well?
- sufficient optimality condition: $\bar{c}_j(\boldsymbol{\pi}) = c_j - \boldsymbol{\pi}^t \mathbf{a}_j \geq 0, \forall j \in J$

Idea for an Algorithm: Explicit Pricing

- this suggests a natural iterative procedure to solve the MP:

$\boxed{\text{solve RMP (with } J'\text{) to optimality}}$

Idea for an Algorithm: Explicit Pricing

- this suggests a natural iterative procedure to solve the MP:

solve RMP (with J') to optimality

π

Idea for an Algorithm: Explicit Pricing

- this suggests a natural iterative procedure to solve the MP:

$$
\begin{tikzcd}
\boxed{\text{solve RMP (with } J'\text{) to optimality}} \arrow[d, bend left, "\boldsymbol{\pi}"] \\
\boxed{\text{compute sign of all } \bar{c}_j(\boldsymbol{\pi}),\ j \in J \setminus J'}
\end{tikzcd}
$$

Idea for an Algorithm: Explicit Pricing

- this suggests a natural iterative procedure to solve the MP:

$J' \leftarrow J' \cup \{j^*\}$

$\boxed{\text{solve RMP (with } J'\text{) to optimality}}$

π

λ_{j^*} if $\bar{c}_{j^*}(\pi) < 0$

$\boxed{\text{compute sign of all } \bar{c}_j(\pi),\ j \in J \setminus J'}$

Idea for an Algorithm: Explicit Pricing

- this suggests a natural iterative procedure to solve the MP:

$J' \leftarrow J' \cup \{j^*\}$

$\boxed{\text{solve RMP (with } J'\text{) to optimality}}$

π

λ_{j^*} if $\bar{c}_{j^*}(\pi) < 0$

stop, if $\bar{c}_j(\pi) \geq 0\ \forall j$ $\longleftarrow$ $\boxed{\text{compute sign of all } \bar{c}_j(\pi),\ j \in J \setminus J'}$

- however, this explicit pricing is *totally out of the question*

i.e., complete enumeration of all variables, these are simply too many

Better Idea: Implicit Pricing

- instead: solve an *auxiliary* optimization problem // implicit enumeration

$$\bar{c}^*(\boldsymbol{\pi}) = \min\{\bar{c}_j(\boldsymbol{\pi}) \mid j \in J\}$$

- this is called the *pricing problem*, *subproblem*, *oracle*, or *column generator*

→ if $\bar{c}^*(\boldsymbol{\pi}) < 0$, we set $J' \leftarrow J' \cup \arg\min_{j \in J}\{\bar{c}_j(\boldsymbol{\pi})\}$
and re-optimize the restricted master problem

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

✎ keep in mind

this identifies a master variable of negative reduced cost or proves that none exists

Better Idea: Implicit Pricing

- this is almost the same as before, with a "tiny detail" changed

$J' \leftarrow J' \cup \{j^*\}$

$\boxed{\text{solve RMP (with } J'\text{) to optimality}}$

λ_{j^*} if $\bar{c}_{j^*} < 0$ $\qquad \pi$

stop, if $\bar{c}_j \geq 0 \ \forall j$ $\longleftarrow$ $\boxed{\text{compute sign of all } \bar{c}_j,\ j \in J \setminus J'}$

Better Idea: Implicit Pricing

- this is almost the same as before, with a “tiny detail” changed

$J' \leftarrow J' \cup \{j^*\}$

solve RMP (with J') to optimality

λ_{j^*} if $\bar{c}_{j^*} < 0$

π

stop, if $\bar{c}_j \geq 0 \ \forall j$ $\leftarrow$ compute sign of ~~all~~ $\bar{c}_j$, $j \in J \setminus J'$

a most negative

Summary: The Column Generation Algorithm

algorithm column generation

input: restricted master problem RMP with an initial set $J' \subseteq J$ of variables;
output: optimal solution $\boldsymbol{\lambda}$ to the master problem MP;

repeat
  solve RMP to optimality, obtain $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$;
  solve pricing problem $\bar{c}^*(\boldsymbol{\pi}) = \min\{\bar{c}_j(\boldsymbol{\pi}) \mid j \in J\}$;
  if $\bar{c}^*(\boldsymbol{\pi}) < 0$ **then**
    $J' \leftarrow J' \cup \{j^*\}$ with $\bar{c}_{j^*}(\boldsymbol{\pi}) = \bar{c}^*(\boldsymbol{\pi})$; // add variable λ_{j^*} to RMP
until $\bar{c}^*(\boldsymbol{\pi}) \geq 0$;

Origins of the Method: Around 1958 in the Western World

A SUGGESTED COMPUTATION FOR MAXIMAL MULTI-COMMODITY NETWORK FLOWS*

L. R. FORD, JR. AND D. R. FULKERSON

The RAND Corporation, Santa Monica, California

A simplex computation for an arc-chain formulation of the maximal multi-commodity network flow problem is proposed. Since the number of variables in this formulation is too large to be dealt with explicitly, the computation treats non-basic variables implicitly by replacing the usual method of determining a vector to enter the basis with several applications of a combinatorial algorithm for finding a shortest chain joining a pair of points in a network.

Management Science, 5(1):97-101, 1958

But already around/before 1951 in the former Soviet Union

Kantorovich and Zalgaller. Calculation of rational cutting of stock, Leningrad, Lenizdat, 1951

Example: Cutting Stock: Restricted Master Problem

- solve (restricted) LP relaxation of Kantorovich-Gilmore-Gomory formulation

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

with $\mathcal{P}' \subseteq \mathcal{P} = \{(a_1, \ldots, a_n)^t \in \mathbb{Z}_+^n \mid \sum_{i=1}^n w_i a_i \leq W\}$

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

// one dual variable per demand

Example: Cutting Stock: Reduced Cost

- reduced cost of λ_p:

$$\bar{c}_p(\boldsymbol{\pi}) = 1 - (\pi_1, \ldots, \pi_n) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{np} \end{pmatrix} \overset{!}{\geq} 0$$

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

- again: explicit enumeration of all patterns impracticable

Example: Cutting Stock: Pricing Problem

- instead: solve auxiliary optimization problem

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

Example: Cutting Stock: Pricing Problem

- instead: solve auxiliary optimization problem

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

$$= \min \quad 1 - \sum_{i=1}^{n} \pi_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i \in \mathbb{Z}_+ \quad i \in [n] \; .$$

Example: Cutting Stock: Pricing Problem

- instead: solve auxiliary optimization problem

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

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

- which is an integer knapsack problem!

Example: Cutting Stock: Pricing Problem

- two cases for the minimum reduced cost $\bar{c}^*(\boldsymbol{\pi}) = \min_{p \in \mathcal{P}} \bar{c}_p(\boldsymbol{\pi})$:

1. $\bar{c}^*(\boldsymbol{\pi}) < 0$,
   that is, $(x_i)_{i \in [n]}$ represents a feasible pattern $p = (a_{ip})_{i \in [n]}$
   $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{p\}$;
   repeat solving the RMP.
2. $\bar{c}^*(\boldsymbol{\pi}) \geq 0$
   proves that there is no negative reduced cost (master variable that corresponds to a) feasible pattern

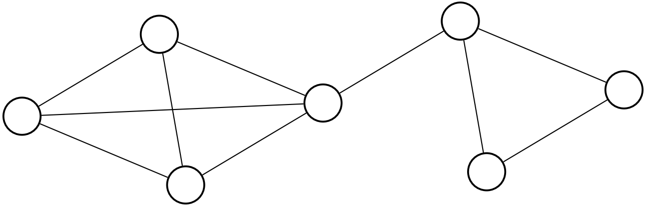
Another Example: Vertex Coloring

Data

$G = (V, E)$ undirected graph

Goal

color all vertices such that adjacent vertices receive different colors, minimizing the number of used colors // like almost all problems we are interested in, this is NP-hard



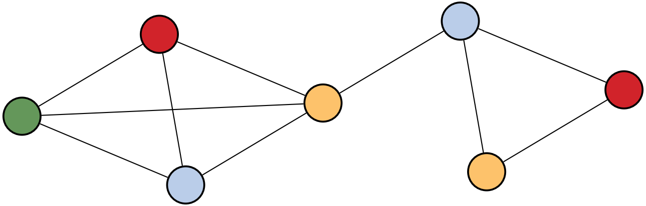
Another Example: Vertex Coloring

Data

$G = (V, E)$ undirected graph

Goal

color all vertices such that adjacent vertices receive different colors, minimizing the number of used colors // like almost all problems we are interested in, this is NP-hard



Vertex Coloring: Textbook Model

- notation: C set of available colors

Vertex Coloring: Textbook Model

- notation: C set of available colors

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

Vertex Coloring: Textbook Model

- notation: C set of available colors

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

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

Vertex Coloring: Textbook Model

- notation: C set of available colors

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

Vertex Coloring: Textbook Model

- notation: C set of available colors

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

Vertex Coloring: Textbook Model

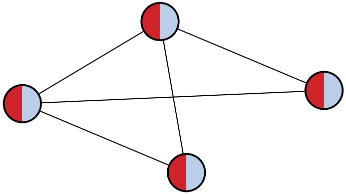
- notation: C set of available colors

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

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

Defects of the Textbook Model

- the LP relaxation is extremely weak



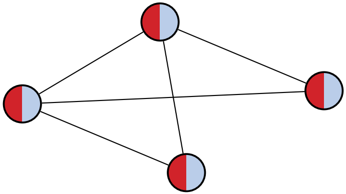
- optimal fractional solution, e.g.,

  $x_{ic_1} = x_{ic_2} = 0.5,\ i \in V$

  $y_{c_1} = y_{c_2} = 0.5$

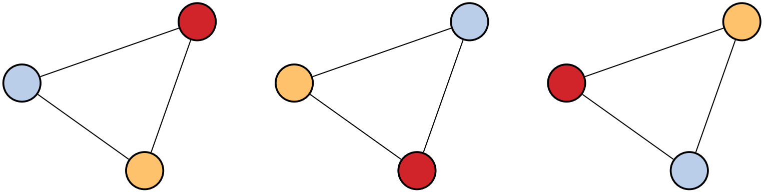
Defects of the Textbook Model

- the LP relaxation is extremely weak



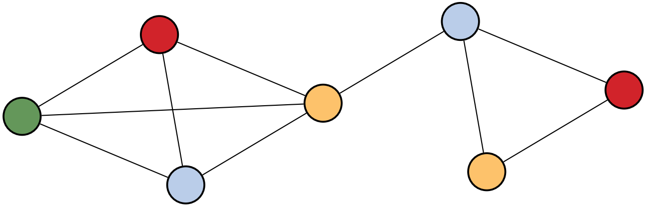
- optimal fractional solution, e.g.,
  $x_{ic_1} = x_{ic_2} = 0.5,\ i \in V$
  $y_{c_1} = y_{c_2} = 0.5$

- *model symmetry* in C: for every feasible x_{ic}, y_c and a permutation $\phi : C \to C$, also $x_{i\phi(c)}$, $y_{\phi(c)}$ is feasible



An Alternative Model based on Color Classes

- observation: every color class forms an *independent/stable set*



- coloring: a partition of the vertex set V into independent sets

An Alternative Model based on Color Classes

- the set $\mathcal{P}$ of (encodings of) all independent sets in G is

$$\mathcal{P} = \left\{ \begin{pmatrix} a_1 \\ \vdots \\ a_{|V|} \end{pmatrix} \in \{0,1\}^{|V|} \mid a_i + a_j \leq 1 \;\; \forall ij \in E \right\}$$

- for each $p \in \mathcal{P}$, denote by $a_{ip} \in \{0,1\}$ whether vertex i is contained in independent set p

Vertex Coloring: Master Problem

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

Vertex Coloring: Master Problem

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}} a_{ip} \lambda_p = 1 \qquad i \in V \qquad \text{// every vertex must be covered}$$

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

Vertex Coloring: Master Problem

$$
\begin{array}{llll}
\min & \displaystyle\sum_{p \in \mathcal{P}} \lambda_p & & \text{// minimimize number of sets used} \\
\text{s.t.} & \displaystyle\sum_{p \in \mathcal{P}} a_{ip} \lambda_p = 1 & i \in V & \text{// every vertex must be covered} \\
& \lambda_p \in \{0,1\} & p \in \mathcal{P} & \text{// do we use independent set } p\text{?}
\end{array}
$$

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

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

Vertex Coloring: Pricing Problem

- the pricing problem looks like

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

Vertex Coloring: Pricing Problem

- the pricing problem looks like

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

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

Vertex Coloring: Pricing Problem

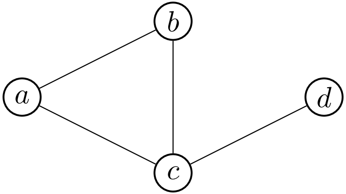
- the pricing problem looks like

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

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

- which is a maximum weight independent set problem!

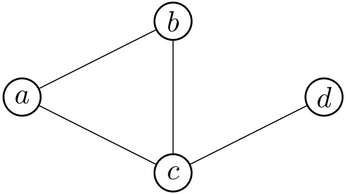
Tiny Numerical Example



$$\mathcal{P} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{a,d\}, \{b,d\}\} \quad \to \quad \text{MP}$$

$$
\begin{array}{lcccccccccccccc}
\min & \lambda_{\{a\}} & + & \lambda_{\{b\}} & + & \lambda_{\{c\}} & + & \lambda_{\{d\}} & + & \lambda_{\{a,d\}} & + & \lambda_{\{b,d\}} & & & \\
\text{s.t.} & \lambda_{\{a\}} & & & & & & & + & \lambda_{\{a,d\}} & & & = & 1 & [\pi_a] \\
 & & & \lambda_{\{b\}} & & & & & & & + & \lambda_{\{b,d\}} & = & 1 & [\pi_b] \\
 & & & & & \lambda_{\{c\}} & & & & & & & = & 1 & [\pi_c] \\
 & & & & & & & \lambda_{\{d\}} & + & \lambda_{\{a,d\}} & + & \lambda_{\{b,d\}} & = & 1 & [\pi_d] \\
 & \lambda_{\{a\}} & , & \lambda_{\{b\}} & , & \lambda_{\{c\}} & , & \lambda_{\{d\}} & , & \lambda_{\{a,d\}} & , & \lambda_{\{b,d\}} & \geq & 0 &
\end{array}
$$

Tiny Numerical Example

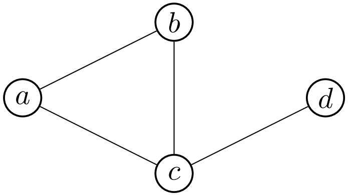


$$\mathcal{P} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{a,d\}, \{b,d\}\} \quad \rightarrow \quad \text{MP}$$

stable sets are encoded by their incidence vectors

$$\begin{array}{llllllllll}
\min & \lambda_{\{a\}} & + \lambda_{\{b\}} & + \lambda_{\{c\}} & + \lambda_{\{d\}} & + \lambda_{\{a,d\}} & + \lambda_{\{b,d\}} & & & \\
\text{s.t.} & \lambda_{\{a\}} & & & & + \lambda_{\{a,d\}} & + 0\lambda_{\{b,d\}} & = & 1 & [\pi_a] \\
 & & \lambda_{\{b\}} & & & & + 1\lambda_{\{b,d\}} & = & 1 & [\pi_b] \\
 & & & \lambda_{\{c\}} & & & + 0\lambda_{\{b,d\}} & = & 1 & [\pi_c] \\
 & & & & \lambda_{\{d\}} & + \lambda_{\{a,d\}} & + 1\lambda_{\{b,d\}} & = & 1 & [\pi_d] \\
 & \lambda_{\{a\}}, & \lambda_{\{b\}}, & \lambda_{\{c\}}, & \lambda_{\{d\}}, & \lambda_{\{a,d\}}, & \lambda_{\{b,d\}} & \geq & 0 &
\end{array}$$

Tiny Numerical Example



$$\mathcal{P}' = \{\{a\}, \{b\}, \{c\}, \{d\}, \{a,d\}, \{b,d\}\} \rightarrow \text{RMP}$$

$$
\begin{array}{rlcr}
\min & \lambda_{\{a\}} + \lambda_{\{b\}} + \lambda_{\{c\}} + \lambda_{\{d\}} + \lambda_{\{a,d\}} + \lambda_{\{b,d\}} & & \\
\text{s.t.} & \lambda_{\{a\}} + \lambda_{\{a,d\}} & = 1 & [\pi_a] \\
 & \lambda_{\{b\}} + \lambda_{\{b,d\}} & = 1 & [\pi_b] \\
 & \lambda_{\{c\}} & = 1 & [\pi_c] \\
 & \lambda_{\{d\}} + \lambda_{\{a,d\}} + \lambda_{\{b,d\}} & = 1 & [\pi_d] \\
 & \lambda_{\{a\}}, \lambda_{\{b\}}, \lambda_{\{c\}}, \lambda_{\{d\}}, \lambda_{\{a,d\}}, \lambda_{\{b,d\}} & \geq 0 &
\end{array}
$$

Tiny Numerical Example

- RMP optimal: $\lambda = \lambda = \lambda = \lambda = 1$, $\pi_a = \pi_b = \pi_c = \pi_d = 1$
- pricing:

$$
\begin{array}{rcccccccccl}
\bar{c}^*(\boldsymbol{\pi}) = \min & 1 - & \pi_a x_a & - & \pi_b x_b & - & \pi_c x_c & - & \pi_d x_d & & \\
& & x_a & + & x_b & & & & & \leq & 1 \\
& & x_a & & & + & x_c & & & \leq & 1 \\
& & & & x_b & + & x_c & & & \leq & 1 \\
& & & & & & x_c & + & x_d & \leq & 1 \\
& & x_a & , & x_b & , & x_c & , & x_d & \in & \{0,1\}
\end{array}
$$

$\Rightarrow$ we see that for the two missing variables $\bar{c} = \bar{c} = -1$

$\Rightarrow$ we add λ or λ to the RMP, and iterate

Tiny Numerical Example

- observe: an optimal (“fractional”) solution to the MP is e.g.,

$$\lambda = \lambda = \lambda = 1$$

- the dual bound we obtain from this solution is 3
- compare: the dual bound we obtain from the original LP relaxation is 1!

$\rightarrow$ it is a coincidence that this solution is integer and the dual bound is tight

Initialization: How to choose an initial $\mathcal{P}'$?

- e.g., for the vertex coloring problem:

$$\mathcal{P}' = \bigcup_{i \in V} \{(a_1, \ldots, a_{|V|})^t\} \text{ with } a_v = \begin{cases} 1 & v = i \\ 0 & v \neq i \end{cases}$$

i.e., one stable set per vertex, consisting of that single vertex

Initialization: How to choose an initial $\mathcal{P}'$?

- e.g., for the vertex coloring problem:

$$\mathcal{P}' = \bigcup_{i \in V} \{(a_1, \ldots, a_{|V|})^t\} \text{ with } a_v = \begin{cases} 1 & v = i \\ 0 & v \neq i \end{cases}$$

  i.e., one stable set per vertex, consisting of that single vertex

- or use a heuristic
- or artificial variables ("phase I")
- or use Farkas pricing

  $\vdots$

Practically Important: Several Pricing Problems

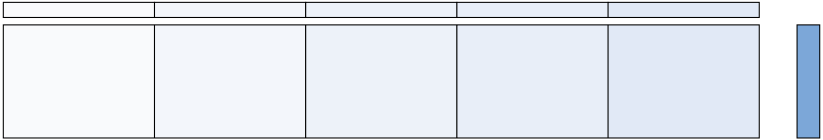
- in applications we often have different, say K many "types" of objects

  // different types of vehicles, containers, material, persons, ...

$$
\begin{aligned}
z^\star_{\mathrm{MP}} = \min \quad & \sum_{k\in[K]} \sum_{j\in J_k} c^k_j \lambda^k_j \\
\text{s.t.} \quad & \sum_{k\in[K]} \sum_{j\in J_k} \mathbf{a}^k_j \lambda^k_j \geq \mathbf{b} \quad [\boldsymbol{\pi}] \\
& \lambda^k_j \geq 0 \qquad \forall k \in [K]\ \forall j \in J_k
\end{aligned}
$$

Practically Important: Several Pricing Problems

- the K classes of variables have their respective own "realms"



$\lambda_p^1,\ p \in \mathcal{P}_1$ $\quad \lambda_p^2,\ p \in \mathcal{P}_2$ $\quad \cdots \quad$ $\lambda_p^K,\ p \in \mathcal{P}_K$

- this requires K pricing problems: $\bar{c}^{k*}(\boldsymbol{\pi}) = \min\{\bar{c}_j^k(\boldsymbol{\pi}) \mid j \in J_k\},\ k \in [K]$
- the optimality condition becomes: $\bar{c}_j^k(\boldsymbol{\pi}) = c_j^k - \boldsymbol{\pi}^t \mathbf{a}_j^k \geq 0,\ \forall k \in [K]\ \forall j \in J_k$

Dual Bounds on the Master Problem Optimum

Interpretation of reduced costs in linear programs in general

- $\bar{c}_j(\boldsymbol{\pi})$ is the potential objective improvement when λ_j is increased by one unit

- assume that we know a κ with $\sum_{j \in J} \lambda_j \leq \kappa$ for any λ-solution

$\Rightarrow$ current objective function value cannot be improved by more than $\kappa \cdot \min_{j \in J} \bar{c}_j(\boldsymbol{\pi})$

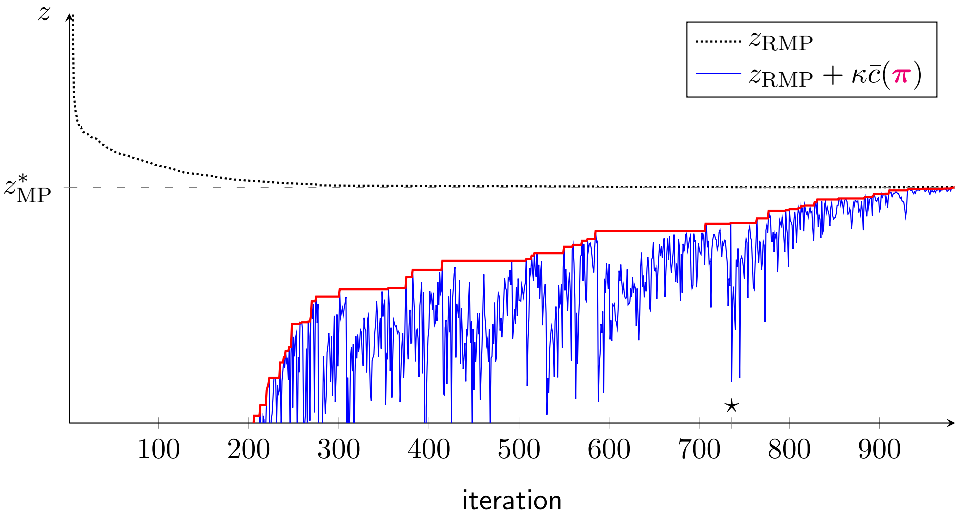
Lemma (*Lagrangian bound*)

Given the optimal values of the RMP and pricing problem, z_{RMP} and $\bar{c}^*(\boldsymbol{\pi})$, then

$$z_{\mathrm{RMP}} + \kappa \cdot \bar{c}^*(\boldsymbol{\pi}) \leq z^*_{\mathrm{MP}}$$

when column generation stops, i.e., when $\bar{c}^* = 0$, the bound becomes tight

Dual Bounds on the Master Problem Optimum



Operations Research | RWTH AACHEN UNIVERSITY

Tailing Off and Early Termination

- *tailing off*: the slow convergence at the end of the column generation process
- given a dual bound on the master optimum, we could stop generating columns when a certain (relative) quality is reached, called *early termination*

Question

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

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation
2.1 Dantzig-Wolfe Reformulation for LPs
2.2 Column Generation
2.3 Multiple Subproblems and Aggregation

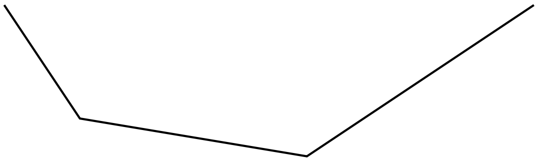
3 Branch-Price-and-Cut

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q\in Q}, \{\mathbf{x}_r\}_{r\in R} \subseteq \mathbb{R}^n$ such that
   $P = \text{conv}(\{\mathbf{x}_q\}_{q\in Q}) + \text{cone}(\{\mathbf{x}_r\}_{r\in R})$ // "P is finitely generated"



- choose $\{\mathbf{x}_q\}_{q\in Q}$ (resp. $\{\mathbf{x}_r\}_{r\in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q\in Q}, \{\mathbf{x}_r\}_{r\in R} \subseteq \mathbb{R}^n$ such that
   $P = \mathrm{conv}(\{\mathbf{x}_q\}_{q\in Q}) + \mathrm{cone}(\{\mathbf{x}_r\}_{r\in R})$ // "P is finitely generated"

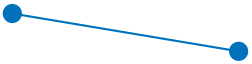
- choose $\{\mathbf{x}_q\}_{q\in Q}$ (resp. $\{\mathbf{x}_r\}_{r\in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q\in Q}, \{\mathbf{x}_r\}_{r\in R} \subseteq \mathbb{R}^n$ such that
   $P = \text{conv}(\{\mathbf{x}_q\}_{q\in Q}) + \text{cone}(\{\mathbf{x}_r\}_{r\in R})$ // "P is finitely generated"



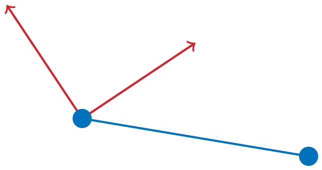
- choose $\{\mathbf{x}_q\}_{q\in Q}$ (resp. $\{\mathbf{x}_r\}_{r\in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q \in Q}, \{\mathbf{x}_r\}_{r \in R} \subseteq \mathbb{R}^n$ such that
   $P = \mathrm{conv}(\{\mathbf{x}_q\}_{q \in Q}) + \mathrm{cone}(\{\mathbf{x}_r\}_{r \in R})$ // "P is finitely generated"



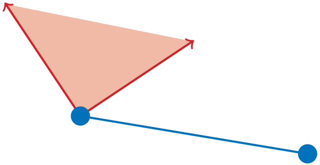
- choose $\{\mathbf{x}_q\}_{q \in Q}$ (resp. $\{\mathbf{x}_r\}_{r \in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q\in Q}, \{\mathbf{x}_r\}_{r\in R} \subseteq \mathbb{R}^n$ such that
   $P = \mathrm{conv}(\{\mathbf{x}_q\}_{q\in Q}) + \mathrm{cone}(\{\mathbf{x}_r\}_{r\in R})$ // "P is finitely generated"



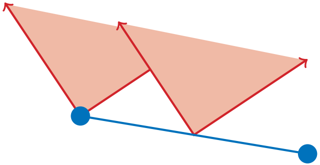
- choose $\{\mathbf{x}_q\}_{q\in Q}$ (resp. $\{\mathbf{x}_r\}_{r\in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q\in Q}, \{\mathbf{x}_r\}_{r\in R} \subseteq \mathbb{R}^n$ such that
   $P = \text{conv}(\{\mathbf{x}_q\}_{q\in Q}) + \text{cone}(\{\mathbf{x}_r\}_{r\in R})$ // "P is finitely generated"



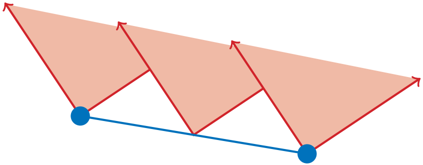
- choose $\{\mathbf{x}_q\}_{q\in Q}$ (resp. $\{\mathbf{x}_r\}_{r\in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q \in Q}, \{\mathbf{x}_r\}_{r \in R} \subseteq \mathbb{R}^n$ such that
   $P = \text{conv}(\{\mathbf{x}_q\}_{q \in Q}) + \text{cone}(\{\mathbf{x}_r\}_{r \in R})$ // "P is finitely generated"



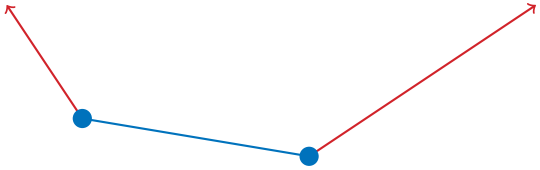
- choose $\{\mathbf{x}_q\}_{q \in Q}$ (resp. $\{\mathbf{x}_r\}_{r \in R}$) as P's extreme points (resp. extreme rays)

Minkowski (1896), Farkas (1902), Weyl (1935)

Outer and Inner Representation of a Polyhedron

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

1. P is a polyhedron
2. There are finite sets $\{\mathbf{x}_q\}_{q\in Q}, \{\mathbf{x}_r\}_{r\in R} \subseteq \mathbb{R}^n$ such that
   $P = \text{conv}(\{\mathbf{x}_q\}_{q\in Q}) + \text{cone}(\{\mathbf{x}_r\}_{r\in R})$ // "P is finitely generated"



- choose $\{\mathbf{x}_q\}_{q\in Q}$ (resp. $\{\mathbf{x}_r\}_{r\in R}$) as P's extreme points (resp. extreme rays)

That's Nice!

- so let's use this!

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

- we will now equivalently reformulate what we call in this context the

original model $$\begin{aligned} z^\star_{\mathrm{LP}} = \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b} \\ & D\mathbf{x} \geq \mathbf{d} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

- we will now equivalently reformulate what we call in this context the

$$\textit{original}\text{ model}\quad z^*_{\mathrm{LP}} = \min \quad \mathbf{c}^t\mathbf{x}$$
$$\text{s. t.}\quad A\mathbf{x} \geq \mathbf{b}$$
$$D\mathbf{x} \geq \mathbf{d}$$
$$\mathbf{x} \geq \mathbf{0}$$

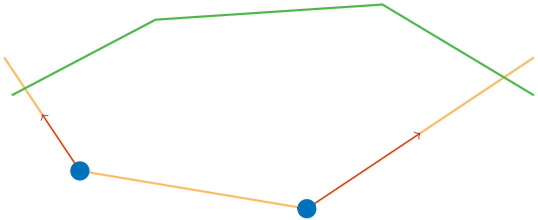
- identify two sets of constraints, typically constraints we know how to deal (well) with and everything else.

e.g., network flow constraints, etc.

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

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

- idea: apply Farkas-Minkowski-Weyl on $X = \{\mathbf{x} \geq \mathbf{0} \mid D\mathbf{x} \geq \mathbf{d}\}$



extreme points $\{\mathbf{x}_q\}_{q \in Q}$, extreme rays $\{\mathbf{x}_r\}_{r \in R}$ of X

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

extreme points $\{\mathbf{x}_q\}_{q\in Q}$, extreme rays $\{\mathbf{x}_r\}_{r\in R}$ of X

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

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

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

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

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

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

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

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

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

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

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

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

Operations Research | RWTH AACHEN UNIVERSITY

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

leads to an *extended* linear program which we call the *Dantzig-Wolfe master problem*

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

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

What did we just do?

- we discovered this nice theorem by Minkowski, Weyl, and Farkas
- we reformulated part of the constraints of an LP according to this theorem

$\rightarrow$ because we can!

- now we have an equivalent LP with a gigantic number of variables

$\rightarrow$ but this doesn't scare us! // because we know column generation

$\Rightarrow$ that makes it quite obvious what comes next ...

The Dantzig-Wolfe Restricted Master Problem

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

Reduced Cost Computation

- for the reduced cost formula we distinguish two cases

→ for λ_q, $q \in Q$:

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

$$= \mathbf{c}^t \mathbf{x}_q - \boldsymbol{\pi}^t A \mathbf{x}_q - \pi_0$$

→ for λ_r, $r \in R$:

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

$$= \mathbf{c}^t \mathbf{x}_r - \boldsymbol{\pi}^t A \mathbf{x}_r$$

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

Dantzig-Wolfe Pricing Problem

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

Dantzig-Wolfe Pricing Problem

- in words: find an extreme point $\mathbf{x}_q, q \in Q$ with minimum $\bar{c}_q$ and/or an extreme ray $\mathbf{x}_r, r \in R$ with minimum $\bar{c}_r$
- to this end, consider an "almost correct" problem $\min_{j \in Q \cup R} \mathbf{c}^t \mathbf{x}_j - \boldsymbol{\pi}^t A \mathbf{x}_j - \pi_0$

  // the objective function (reduced cost) is off by a constant $-\pi_0$ for rays

Dantzig-Wolfe Pricing Problem

- in words: find an extreme point $\mathbf{x}_q, q \in Q$ with minimum $\bar{c}_q$ and/or an extreme ray $\mathbf{x}_r, r \in R$ with minimum $\bar{c}_r$
- to this end, consider an "almost correct" problem $\min_{j \in Q \cup R} \mathbf{c}^t \mathbf{x}_j - \boldsymbol{\pi}^t A \mathbf{x}_j - \pi_0$

  // the objective function (reduced cost) is off by a constant $-\pi_0$ for rays

- Q and R index the extreme points/extreme rays of $\{\mathbf{x} \geq \mathbf{0} \mid D\mathbf{x} \geq \mathbf{d}\}$!

  // and we know how to obtain these extreme points/rays

Dantzig-Wolfe Pricing Problem

- in words: find an extreme point $\mathbf{x}_q, q \in Q$ with minimum $\bar{c}_q$ and/or an extreme ray $\mathbf{x}_r, r \in R$ with minimum $\bar{c}_r$
- to this end, consider an "almost correct" problem $\min_{j \in Q \cup R} \mathbf{c}^t \mathbf{x}_j - \boldsymbol{\pi}^t A \mathbf{x}_j - \pi_0$

  // the objective function (reduced cost) is off by a constant $-\pi_0$ for rays
- leading to the *Dantzig-Wolfe pricing problem*

beauty alert!

$$
\begin{aligned}
z^*_{\mathrm{PP}} \quad = \quad \min \quad & (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \\
\text{s.t.} \quad & D\mathbf{x} \geq \mathbf{d} \\
& \mathbf{x} \geq \mathbf{0}
\end{aligned}
$$

- Q and R index the extreme points/extreme rays of $\{\mathbf{x} \geq \mathbf{0} \mid D\mathbf{x} \geq \mathbf{d}\}$!

  // and we know how to obtain these extreme points/rays
- the pricing problem is again a linear program

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \left\{ (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d} \right\}$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\mathrm{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d}\}$

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

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x}\geq\mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $\mathbf{x}_{r^*}, r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$
   // the precise value of $\bar{c}_{r^*}$ is not relevant in this case, so we “accept” the wrong objective function

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\mathrm{PP}} = \min_{\mathbf{x}\geq\mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\mathrm{PP}} = -\infty \Rightarrow$ we identified an extreme ray $\mathbf{x}_{r^*}, r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$
   // the precise value of $\bar{c}_{r^*}$ is not relevant in this case, so we "accept" the wrong objective function

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

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $\mathbf{x}_{r^*}, r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t \mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$
   // the precise value of $\bar{c}_{r^*}$ is not relevant in this case, so we "accept" the wrong objective function

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

Dantzig-Wolfe Pricing Problem

three cases for $z^\star_{\mathrm{PP}} = \min_{\mathbf{x}\geq\mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^\star_{\mathrm{PP}} = -\infty \Rightarrow$ we identified an extreme ray $\mathbf{x}_{r^*}, r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$
   // the precise value of $\bar{c}_{r^*}$ is not relevant in this case, so we "accept" the wrong objective function

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

3. $0 \leq z^\star_{\mathrm{PP}}$

Dantzig-Wolfe Pricing Problem

three cases for $z^\star_{\text{PP}} = \min_{\mathbf{x}\geq\mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^\star_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $\mathbf{x}_{r^*}, r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\to$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$
   // the precise value of $\bar{c}_{r^*}$ is not relevant in this case, so we "accept" the wrong objective function

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

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

Projecting back to the Original Variables

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

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

- this projection becomes interesting when we are interested in integer solutions

Wrap-Up

- we Dantzig-Wolfe reformulated a subset of constraints of a linear program
- these constraints are exactly those that appear in the pricing problem
- feasible solutions to these constraints "define" the meaning of the master variables
- this "effect" becomes even more visible when working with integer programs

✎ keep in mind

the variables (and constraints) of the pricing problem are from the original LP

Block-Diagonal Structure

- many models in practice have a *block-diagonal structure* // we know this already

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

- K bins, K colors, K vehicles, K *blocks*, ...

Block-Diagonal Structure: Reformulate the Block Constraints

- in a Dantzig-Wolfe context, the constraints

$$\begin{pmatrix} A_1 & A_2 & \dots & A_K \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{pmatrix} \geq \mathbf{b} \qquad // =: A\mathbf{x} \geq \mathbf{b}$$

are "complicating" because they involve *all variables*, whereas

$$\begin{pmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_K \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{pmatrix} \geq \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_K \end{pmatrix} \qquad // =: D\mathbf{x} \geq \mathbf{d}$$

are "easier" since they decompose into *independent subsystems*

Block-Diagonal Structure: Dantzig-Wolfe Reformulation

- key idea: reformulate each $X_k = \{\mathbf{x}^k \geq \mathbf{0} \mid D_k\mathbf{x}^k \geq \mathbf{d}_k\}$ *individually*
- use extreme points $\{\mathbf{x}_q^k\}_{q\in Q_k}$ and extreme rays $\{\mathbf{x}_r^k\}_{r\in R_k}$ of X_k
- like before, express every $\mathbf{x}^k \in X_k$, $k \in [K]$, as

$$
\begin{aligned}
\mathbf{x}^k = \sum_{q\in Q_k} \lambda_q^k \mathbf{x}_q^k + \sum_{r\in R_k} \lambda_r^k \mathbf{x}_r^k & & \\
\sum_{q\in Q_k} \lambda_q^k &= 1 & \\
\lambda_q^k &\geq 0 & q \in Q_k \\
\lambda_r^k &\geq 0 & r \in R_k
\end{aligned}
$$

- and substitute this $\mathbf{x}^k \in X_k$ in $\sum_{k=1}^{K} A_k\mathbf{x}^k \geq \mathbf{b}$ and $\sum_{k=1}^{K} \mathbf{c}_k^t \mathbf{x}^k$.

Block-Diagonal Structure: Development of the MP

- substitution of $\mathbf{x}^k \in X_k$ in $\sum_{k=1}^{K} A_k \mathbf{x}^k \geq \mathbf{b}$ and $\sum_{k=1}^{K} \mathbf{c}_k^t \mathbf{x}^k$ yields

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K} \mathbf{c}_k^t \left(\sum_{q \in Q_k} \lambda_q^k \mathbf{x}_q^k + \sum_{r \in R_k} \lambda_r^k \mathbf{x}_r^k \right) & & \\
\text{s.t.} \quad & \sum_{k=1}^{K} A_k \left(\sum_{q \in Q_k} \lambda_q^k \mathbf{x}_q^k + \sum_{r \in R_k} \lambda_r^k \mathbf{x}_r^k \right) \geq \mathbf{b} & & \\
& \sum_{q \in Q_k} \lambda_q^k = 1 & & k \in [K] \\
& \lambda_q^k \geq 0 & & k \in [K],\ q \in Q_k \\
& \lambda_r^k \geq 0 & & k \in [K],\ r \in R_k
\end{aligned}
$$

Block-Diagonal Structure: Development of the MP

- substitution of $\mathbf{x}^k \in X_k$ in $\sum_{k=1}^{K} A_k \mathbf{x}^k \geq \mathbf{b}$ and $\sum_{k=1}^{K} \mathbf{c}_k^t \mathbf{x}^k$ yields

$$
\begin{array}{llll}
\min & \sum_{k=1}^{K} \sum_{q \in Q_k} \lambda_q^k \mathbf{c}_k^t \mathbf{x}_q^k + \sum_{k=1}^{K} \sum_{r \in R_k} \lambda_r^k \mathbf{c}_k^t \mathbf{x}_r^k & & \\
\text{s.t.} & \sum_{k=1}^{K} \sum_{q \in Q_k} \lambda_q^k A_k \mathbf{x}_q^k + \sum_{k=1}^{K} \sum_{r \in R_k} \lambda_r^k A_k \mathbf{x}_r^k & \geq \mathbf{b} & \\
& \sum_{q \in Q_k} \lambda_q^k & = 1 & k \in [K] \\
& \lambda_q^k & \geq 0 & k \in [K],\ q \in Q_k \\
& \lambda_r^k & \geq 0 & k \in [K],\ r \in R_k
\end{array}
$$

Block-Diagonal Structure: Development of the MP

- substitution of $\mathbf{x}^k \in X_k$ in $\sum_{k=1}^{K} A_k \mathbf{x}^k \geq \mathbf{b}$ and $\sum_{k=1}^{K} \mathbf{c}_k^t \mathbf{x}^k$ yields

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K} \sum_{q \in Q_k} \lambda_q^k \underbrace{\mathbf{c}_k^t \mathbf{x}_q^k}_{=:c_q^k} + \sum_{k=1}^{K} \sum_{r \in R_k} \lambda_r^k \underbrace{\mathbf{c}_k^t \mathbf{x}_r^k}_{=:c_r^k} \\
\text{s.t.} \quad & \sum_{k=1}^{K} \sum_{q \in Q_k} \lambda_q^k \underbrace{A_k \mathbf{x}_q^k}_{=:\mathbf{a}_q^k} + \sum_{k=1}^{K} \sum_{r \in R_k} \lambda_r^k \underbrace{A_k \mathbf{x}_r^k}_{=:\mathbf{a}_r^k} \geq \mathbf{b} \\
& \sum_{q \in Q_k} \lambda_q^k = 1 \qquad k \in [K] \\
& \lambda_q^k \geq 0 \qquad k \in [K],\ q \in Q_k \\
& \lambda_r^k \geq 0 \qquad k \in [K],\ r \in R_k
\end{aligned}
$$

Block-Diagonal Structure: Development of the MP

- we arrive, again, at the *Dantzig-Wolfe master problem*

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K} \sum_{q \in Q_k} c_q^k \lambda_q^k + \sum_{k=1}^{K} \sum_{r \in R_k} c_r^k \lambda_r^k \\
\text{s.t.} \quad & \sum_{k=1}^{K} \sum_{q \in Q_k} \mathbf{a}_q^k \lambda_q^k + \sum_{k=1}^{K} \sum_{r \in R_k} \mathbf{a}_r^k \lambda_r^k \geq \mathbf{b} && [\boldsymbol{\pi}] \\
& \sum_{q \in Q_k} \lambda_q^k = 1 && [\pi_0^k] && k \in [K] \\
& \lambda_q^k \geq 0 && && k \in [K],\ q \in Q_k \\
& \lambda_r^k \geq 0 && && k \in [K],\ r \in R_k
\end{aligned}
$$

Block-Diagonal Structure: Multiple Pricing Problems

- we now have K *Dantzig-Wolfe pricing problems*

$$\min_{j \in Q_k \cup R_k} \mathbf{c}_k^t \mathbf{x}_j^k - \boldsymbol{\pi}^t A_k \mathbf{x}_j^k - \pi_0^k$$

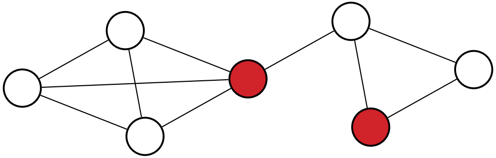
- which, again, we solve as

$$\begin{array}{rcll} z^*_{\mathrm{PP},k} & = & \min & (\mathbf{c}_k^t - \boldsymbol{\pi}^t A_k)\mathbf{x}^k - \pi_0^k \\ & & \text{s. t.} & D_k \mathbf{x}^k \geq \mathbf{d}_k \\ & & & \mathbf{x}^k \geq \mathbf{0} \end{array}$$

- column generation stops when $0 \leq z^*_{\mathrm{PP},k} \ \forall k \in [K]$

(Not so) special Case: Aggregation of Identical Subproblems

- if, e.g., we perform a DW reformulation on the vertex coloring textbook model

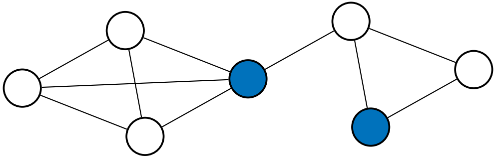


- we arrive at stable sets in many different colors and master constraints

$$\sum_{p \in \mathcal{P}_1 : i \in p} \lambda_p^1 \qquad = 1 \quad i \in V$$

(Not so) special Case: Aggregation of Identical Subproblems

- if, e.g., we perform a DW reformulation on the vertex coloring textbook model

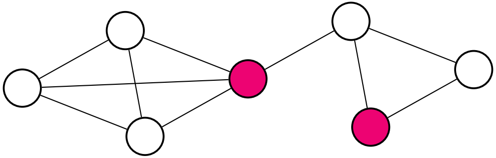


- we arrive at stable sets in many different colors and master constraints

$$\sum_{p \in \mathcal{P}_1 : i \in p} \lambda_p^1 + \sum_{p \in \mathcal{P}_2 : i \in p} \lambda_p^2 = 1 \quad i \in V$$

(Not so) special Case: Aggregation of Identical Subproblems

- if, e.g., we perform a DW reformulation on the vertex coloring textbook model

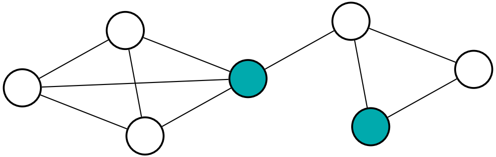


- we arrive at stable sets in many different colors and master constraints

$$\sum_{p\in\mathcal{P}_1:i\in p} \lambda_p^1 + \sum_{p\in\mathcal{P}_2:i\in p} \lambda_p^2 + \sum_{p\in\mathcal{P}_3:i\in p} \lambda_p^3 = 1 \quad i \in V$$

(Not so) special Case: Aggregation of Identical Subproblems

- if, e.g., we perform a DW reformulation on the vertex coloring textbook model



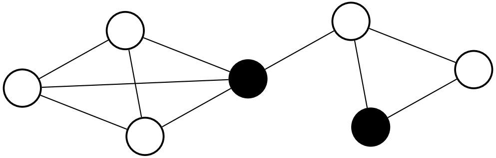
- we arrive at stable sets in many different colors and master constraints

$$\sum_{p \in \mathcal{P}_1 : i \in p} \lambda_p^1 + \sum_{p \in \mathcal{P}_2 : i \in p} \lambda_p^2 + \sum_{p \in \mathcal{P}_3 : i \in p} \lambda_p^3 + \cdots + \sum_{p \in \mathcal{P}_{|C|} : i \in p} \lambda_p^{|C|} = 1 \quad i \in V$$

- but the stable sets are "all the same!" (and pricing problems are "the same")

(Not so) special Case: Aggregation of Identical Subproblems

- if, e.g., we perform a DW reformulation on the vertex coloring textbook model



- we arrive at stable sets in many different colors and master constraints

$$\sum_{p\in\mathcal{P}:i\in p} \lambda_p = \sum_{p\in\mathcal{P}_1:i\in p} \lambda_p^1 + \sum_{p\in\mathcal{P}_2:i\in p} \lambda_p^2 + \sum_{p\in\mathcal{P}_3:i\in p} \lambda_p^3 + \cdots + \sum_{p\in\mathcal{P}_{|C|}:i\in p} \lambda_p^{|C|} = 1 \quad i \in V$$

- but the stable sets are "all the same!" (and pricing problems are "the same")

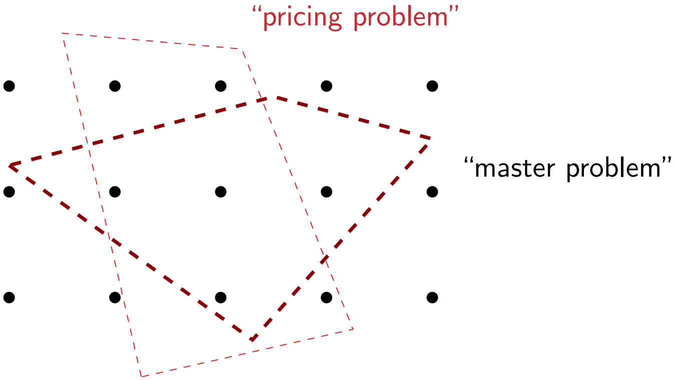
$\Rightarrow$ aggregate $\lambda_p = \lambda_p^1 + \lambda_p^2 + \cdots + \lambda_p^{|C|} \quad p \in \mathcal{P}$ // colorless representation
and use only one "colorless" pricing problem

Mark my Words!

- even if you neva eva DW reformulate an IP in your lives, this is useful stuff

Dantzig-Wolfe and Column Generation for LPs: Pictorially

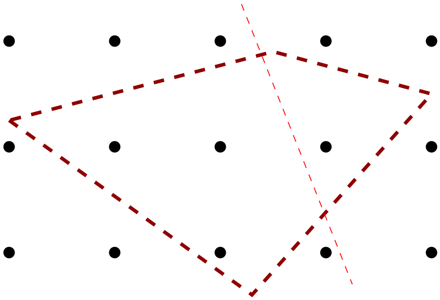
$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \cap \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$



Dantzig-Wolfe and Column Generation for LPs: Pictorially

$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$

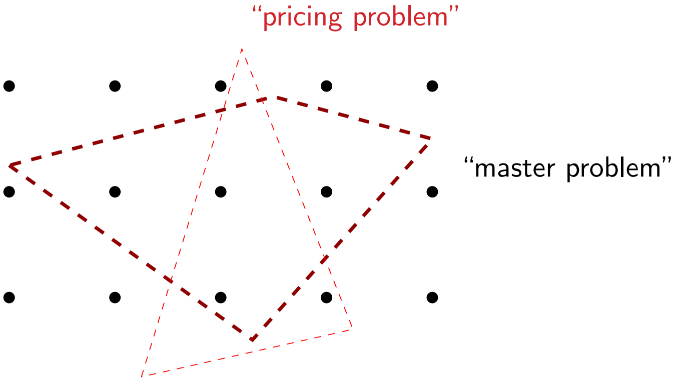
"pricing problem"



"master problem"

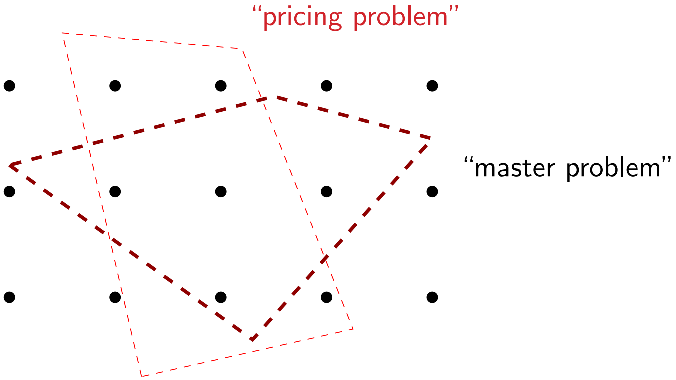
Dantzig-Wolfe and Column Generation for LPs: Pictorially

$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$



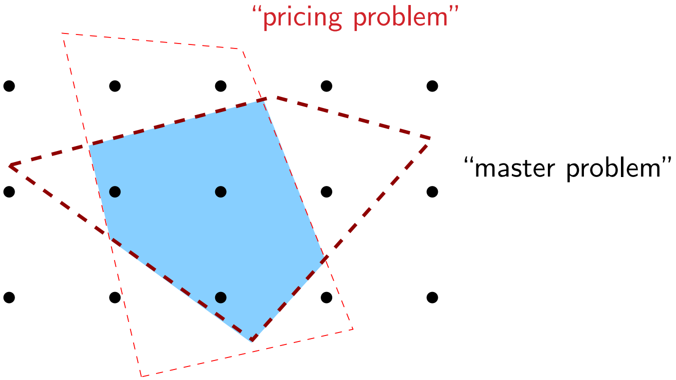
Dantzig-Wolfe and Column Generation for LPs: Pictorially

$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$



Dantzig-Wolfe and Column Generation for LPs: Pictorially

$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$



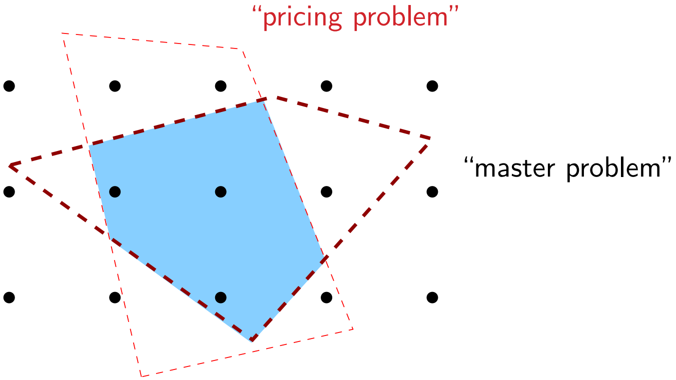
- not tighter than standard LP relaxation

Briefly Pause

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

Dantzig-Wolfe Reformulation for IPs: Pictorially

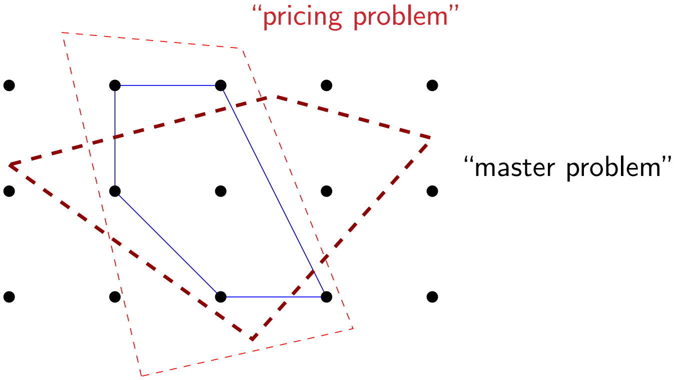
$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$



- not tighter than standard LP relaxation

Dantzig-Wolfe Reformulation for IPs: Pictorially

$$\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$

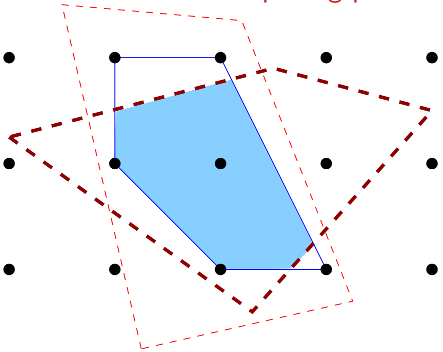


- for integer programs: partial convexification $\operatorname{conv}\{\mathbf{x} \in \mathbb{Z}^n \mid D\mathbf{x} \geq \mathbf{d}\}$

Dantzig-Wolfe Reformulation for IPs: Pictorially

$$\color{red}{\{\mathbf{x} \in \mathbb{Q}^n \mid D\mathbf{x} \geq \mathbf{d}\}} \;\cap\; \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \geq \mathbf{b}\}$$

"pricing problem"



"master problem"

- for integer programs: partial convexification, possibly stronger

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation

3 Branch-Price-and-Cut

3.1 Cutting Planes

3.2 Branching

Strong and Stronger

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

skip to branching

Cutting Planes in the Original Variables

- let us assume that we know a set of cutting planes $F\mathbf{x} \geq \mathbf{f}$ for our original IP

$$
\begin{array}{rrrcl}
z^*_{\mathrm{IP}} := & \min & \mathbf{c}^t\mathbf{x} & & \\
& \text{s.t.} & A\mathbf{x} & \geq & \mathbf{b} \\
& & D\mathbf{x} & \geq & \mathbf{d} \\
& & F\mathbf{x} & \geq & \mathbf{f} \\
& & \mathbf{x} & \in & \mathbb{Z}^n_+
\end{array}
$$

- how do they present themselves in the master problem?

DW Reformulated Cutting Planes appear in the Master

- cuts $F\mathbf{x} \geq \mathbf{f}$ are treated in the same way as $A\mathbf{x} \geq \mathbf{b}$

$$
\begin{array}{llllll}
\min & \sum_{q \in Q} c_q \lambda_q & + & \sum_{r \in R} c_r \lambda_r & & \\
\text{s.t.} & \sum_{q \in Q} \mathbf{a}_q \lambda_q & + & \sum_{r \in R} \mathbf{a}_r \lambda_r & \geq \mathbf{b} & [\boldsymbol{\pi}] \\
 & \sum_{q \in Q} \underbrace{F\mathbf{x}_q}_{=:\mathbf{f}_q} \lambda_q & + & \sum_{r \in R} \underbrace{F\mathbf{x}_r}_{=:\mathbf{f}_r} \lambda_r & \geq \mathbf{f} & [\boldsymbol{\alpha}] \\
 & \sum_{q \in Q} \lambda_q & & & = 1 & [\pi_0] \\
 & \lambda_q & & & \geq 0 & q \in Q \\
 & & & \lambda_r & \geq 0 & r \in R
\end{array}
$$

Small Modifications in the Pricing Problem

- the cuts' dual variables $\boldsymbol{\alpha}$ impact the reduced cost calculation

$$
\begin{array}{lrcl}
\min & (\mathbf{c}^t - \boldsymbol{\pi}^t A - \boldsymbol{\alpha}^t F)\mathbf{x} - \pi_0 & & \\
\text{s.t.} & D\mathbf{x} & \geq & \mathbf{d} \\
 & \mathbf{x} & \in & \mathbb{Z}^n_+
\end{array}
$$

- the pricing problem's domain formally stays the same!
  // "the pricing problem structure does not change"

$\rightarrow$ specialized algorithms for the pricing problem may still work

- from a solution $\mathbf{x}^*$ to the pricing problem one computes the cuts' coefficients in the master problem as usual as $F\mathbf{x}^*$ // the cuts are "lifted"

Algorithmic Modifications: Pricing and Cutting

initialize the RMP as usual
loop
    solve the MP to optimality via column generation to obtain $\boldsymbol{\lambda}^*$;
    project $\boldsymbol{\lambda}^*$ back to original variables $\mathbf{x}^*$;
    call separation algorithms on $\mathbf{x}^*$;
    if *this produces a cut* $\mathbf{f}^t\mathbf{x} \geq f_0$ **then**
        add $\sum_{q \in Q'} \mathbf{f}^t\mathbf{x}_q\lambda_q + \sum_{r \in R'} \mathbf{f}^t\mathbf{x}_r\lambda_r \geq f_0$ to the RMP with dual variable α;
        respect α in objective function of the pricing problem;
    else
        break;

Experimental Strength of Cutting Planes in Original Variables

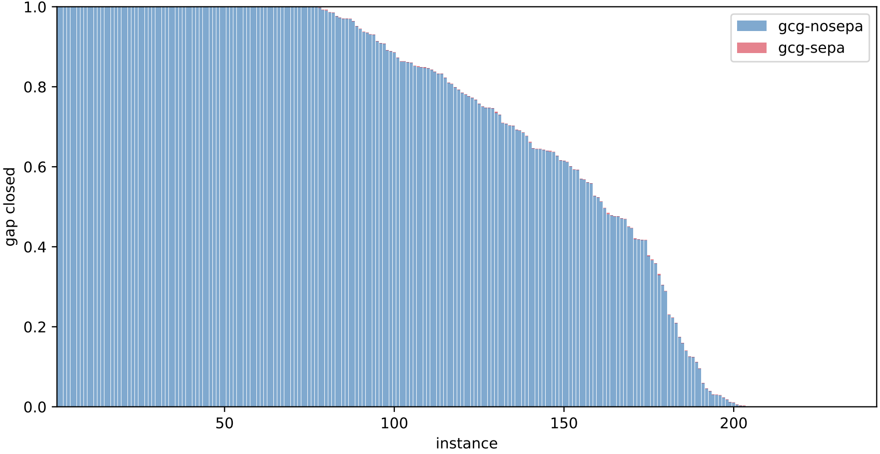
- we performed an experiment on many (mixed) integer programs ("instances"); for each instance compute the integrality gap $(z^*_{\mathrm{IP}} - z^*_{\mathrm{LP}})/z^*_{\mathrm{LP}}$; then report

$\rightarrow$ the portion of the gap that is closed by the DW reformulation

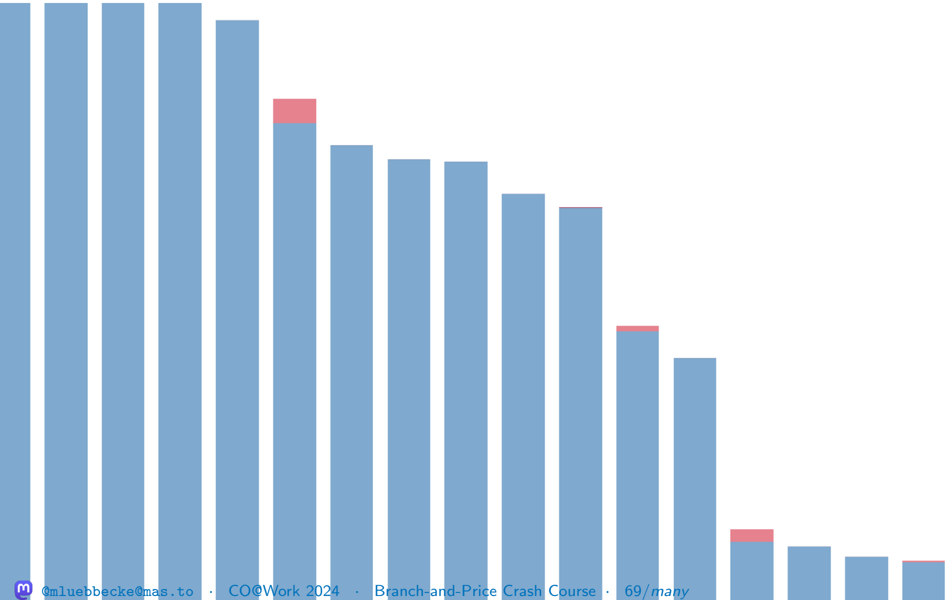
$\rightarrow$ the additional gap closed by generic cuts // those, SCIP can separate

- details in the Ph.D. thesis by Jonas Witt (2019)

Experimental Strength of Cutting Planes in Original Variables



Experimental Strength of Cutting Planes in Original Variables



Experimental Strength of Cutting Planes from Original

- attempt of an interpretation: DW reformulation is so strong that generic cutting planes (from original) are already "implied" // what we observed for vertex coloring
- *a formal proof of such results is an open research topic*
- partial answers in the Ph.D. thesis by Jonas Witt (2019)
- still, in practice, adding (problem specific) cuts appears to be indispensable for a good performance

Where we are

- it is not *either* reformulation *or* cutting planes

- the large body of literature on cutting planes can be combined with Dantzig-Wolfe

$\rightarrow$ usually, only the objective function of the pricing problem needs adaptation

// when cuts involve zero cost variables of the pricing problem, constraints may change

- empirically, most strengthening is to be expected from problem specific cuts

Cutting Planes in the Master Variables

- assuming integer master variables, // which we can always do
  we want to formulate cuts directly on the λ-variables
- this is also the case when the MP is stated as a "pattern based model," not arriving as a DW reformulation

- challenge: when these cuts don't stem from a *counterpart* in original variables, how can we know their coefficients in the pricing problem?

$\rightarrow$ we will have to construct a counterpart in extended original variables!

Cutting Planes in the Master Variables

- the master problem with cuts in the $\boldsymbol{\lambda}$-variables reads

$$
\begin{aligned}
\min \quad & \sum_{q\in Q} c_q\lambda_q && + \sum_{r\in R} c_r\lambda_r && && \\
\text{s.t.} \quad & \sum_{q\in Q} \mathbf{a}_q\lambda_q && + \sum_{r\in R} \mathbf{a}_r\lambda_r && \geq \mathbf{b} && [\boldsymbol{\pi}] \\
& \sum_{q\in Q} \mathbf{g}_q\lambda_q && + \sum_{r\in R} \mathbf{g}_r\lambda_r && \geq \mathbf{h} && [\boldsymbol{\beta}] \\
& \sum_{q\in Q} \lambda_q && && = 1 && [\pi_0] \\
& \lambda_q && && \geq 0 && q \in Q \\
& && \lambda_r && \geq 0 && r \in R
\end{aligned}
$$

We restrict ourselves to Rank-1 Inequalities

- let us consider *rank-1 inequalities*, i.e., cut coefficients $\mathbf{g}_j$ depend only on $\mathbf{a}_j$: $\mathbf{g}_j = g(\mathbf{a}_j) = g(A\mathbf{x}_j)$

$$\sum_{q\in Q} \mathbf{a}_q \lambda_q + \sum_{r\in R} \mathbf{a}_r \lambda_r \geq \mathbf{b} \quad [\boldsymbol{\pi}]$$
$$\sum_{q\in Q} \mathbf{g}_q \lambda_q + \sum_{r\in R} \mathbf{g}_r \lambda_r \geq \mathbf{h} \quad [\boldsymbol{\beta}]$$

- the cut coefficients $\mathbf{g}_j$ impact the reduced cost computation:

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

An Extended Original Problem...

- *if we could* express the dependency $\mathbf{y} = g(A\mathbf{x}_j)$ with *linear* constraints in the original variables $\mathbf{x}$ and potentially *additional original* variables $\mathbf{y}$,
- then, our master problem with the cuts we formulated in the $\boldsymbol{\lambda}$-variables would arrive by a Dantzig-Wolfe reformulation of

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

... yields an Extended Pricing Problem

- this gives the extended pricing problem

$$\begin{array}{lrcl} \min & \mathbf{c}^t\mathbf{x} - \boldsymbol{\pi}^t A\mathbf{x} - \boldsymbol{\beta}^t\mathbf{y} - \pi_0 & & \\ \text{s.t.} & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{y} & = & g(A\mathbf{x}) \\ & \mathbf{x} & \in & \mathbb{Z}^n_+ \end{array}$$

- which is again a (mixed) integer program
- Desaulniers, Desrosiers, and Spoorendonk (2011) extend these considerations to higher rank inequalities

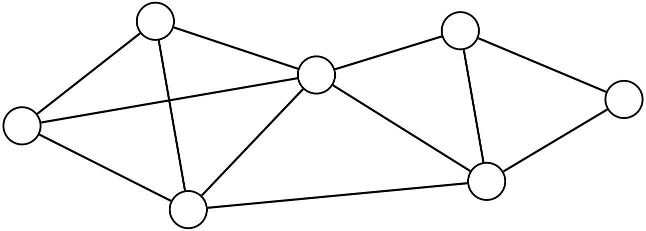
Example: Edge Coloring

Data

undirected graph $G = (V, E)$

Goal

color all edges such that incident edges receive different colors;
minimize the number of used colors



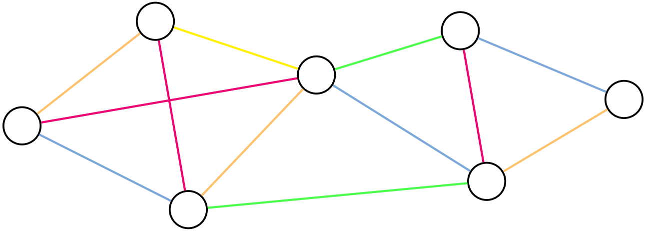
Example: Edge Coloring

Data

undirected graph $G = (V, E)$

Goal

color all edges such that incident edges receive different colors;
minimize the number of used colors



// note: Vizing's theorem states that Δ or $\Delta + 1$ colors suffice, where Δ is the maximum degree in G

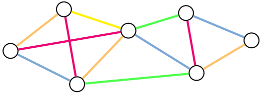
Edge Coloring: A Compact Integer Program

$$
\begin{aligned}
\chi'(G) = \min \quad & \sum_{c \in C} y_c && && \text{// minimize number of used colors} \\
\text{s.t.} \quad & \sum_{c \in C} x_{ec} = 1 && e \in E && \text{// color each edge} \\
& \sum_{e \in \delta(i)} x_{ec} \le y_c && i \in V,\ c \in C && \text{// avoid conflicts} \\
& x_{ec} \in \{0,1\} && e \in E,\ c \in C && \text{// color edge } e \text{ with } c\text{?} \\
& y_c \in \{0,1\} && c \in C && \text{// do we use color } c\text{?}
\end{aligned}
$$

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

Edge Coloring: A Set Partitioning Formulation

- observation: an edge coloring partitions E into matchings



$\Rightarrow$ Nemhauser & Park (1991) formulate a set partitioning model

// this is the aggregated DW reformulation of the previous original IP

$$\begin{aligned} \min \quad & \sum_{j \in J} \lambda_j \\ \text{s.t.} \quad & \sum_{j \in J} \mathbf{a}_j \lambda_j = \mathbf{1} \\ & \lambda_j \in \{0,1\} \qquad j \in J \end{aligned}$$

- with the set J of all matchings in G; the incidence vector $\mathbf{a}_j$ of matching $j \in J$

Edge Coloring: A Set Partitioning Formulation

- solve the LP relaxation by column generation

$$\begin{aligned} \min \quad & \sum_{j \in J} \lambda_j \\ \text{s.t.} \quad & \sum_{j \in J} \mathbf{a}_j \lambda_j = \mathbf{1} & [\boldsymbol{\pi} \text{ free}] \\ & \lambda_j \geq 0 & j \in J \end{aligned}$$

- the pricing problem is

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

Edge Coloring: A Set Partitioning Formulation

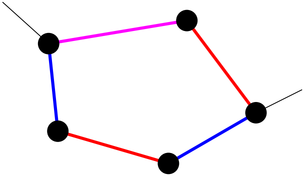
- solve the LP relaxation by column generation

$$
\begin{array}{llcll}
\min & \sum_{j \in J} \lambda_j & & & \\
\text{s.t.} & \sum_{j \in J} \mathbf{a}_j \lambda_j & = & \mathbf{1} & [\boldsymbol{\pi} \text{ free}] \\
 & \lambda_j & \geq & 0 & j \in J
\end{array}
$$

- the pricing problem is

$$
\min \left\{ 1 - \sum_{e \in E} \pi_e x_e \mid \sum_{e \in \delta(i)} x_e \leq 1,\ i \in V,\ x_e \in \{0,1\},\ e \in E \right\}
$$

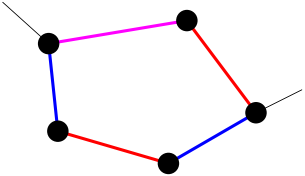
Odd Circuit Cuts



- consider an *odd circuit* C in G

$\rightarrow$ we need at least *three matchings* to cover C

Odd Circuit Cuts



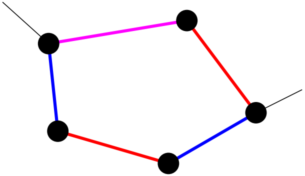
- consider an *odd circuit* C in G

$\rightarrow$ we need at least *three matchings* to cover C

- the *odd circuit cut* derived from C is

$$\sum_{j \in J : j \cap C \neq \emptyset} \lambda_j \geq 3$$

Odd Circuit Cuts



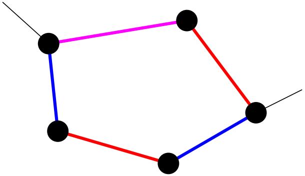
- consider an *odd circuit* C in G

$\rightarrow$ we need at least *three matchings* to cover C

- the *odd circuit cut* derived from C is in the master problem

$$\sum_{j \in J} g(\mathbf{a}_j)\lambda_j \; = \sum_{j \in J : j \cap C \neq \emptyset} \lambda_j \;\geq\; 3 \quad [\beta_C \geq 0]$$

Odd Circuit Cuts



- consider an *odd circuit* C in G

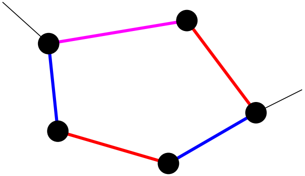
$\rightarrow$ we need at least *three matchings* to cover C

- the *odd circuit cut* derived from C is in the master problem

$$\sum_{j \in J} g(\mathbf{a}_j)\lambda_j \;=\; \sum_{j \in J : j \cap C \neq \emptyset} \lambda_j \;\geq\; 3 \quad [\beta_C \geq 0]$$

- we use a new binary variable $y_C := g(\mathbf{a}_j) = 1 \iff j$ intersects C

Odd Circuit Cuts



- consider an *odd circuit* C in G

$\rightarrow$ we need at least *three matchings* to cover C

- the *odd circuit cut* derived from C is in the master problem

$$\sum_{j\in J} g(\mathbf{a}_j)\lambda_j \;=\; \sum_{j\in J:j\cap C\neq\emptyset} \lambda_j \;\geq\; 3 \quad [\beta_C \geq 0]$$

- we use a new binary variable $y_C := g(\mathbf{a}_j) = 1 \iff j$ intersects C

- this leads to an extended pricing problem:

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

Odd Circuit Cuts: Extended Pricing Problem

- why is this extended pricing problem correct?
  // = why does it produce the correct coefficient in the cut?

$$
\begin{array}{llrcll}
\min & 1 - \sum_{e\in E} \pi_e x_e - \beta_C y_C & & & & \\
\text{s.t.} & & \sum_{e\in\delta(i)} x_e & \leq & 1 & i \in V \\
& & y_C & \leq & \sum_{e\in C} x_e & \\
& & x_e & \in & \{0,1\} & e \in E \\
& & y_C & \in & \{0,1\} &
\end{array}
$$

- "$y_C = 1 \Rightarrow j$ intersects C" is enforced, but not the converse
- however, since $\beta_C \geq 0$ there is an incentive to set $y_C = 1$

$\Rightarrow$ *at optimality*, $y_C = 1 \iff j$ intersects C

Comparing the Strength of Cutting Planes

- cuts on the original are the *special case* $y = g(A\mathbf{x}) = Fx$

  // the cut can be expressed as a linear function of the original variables, no extra $\mathbf{y}$ needed

$\Rightarrow$ master cuts are at least as strong as original cuts

- in order to derive a master cut from the original model one may need additional variables and constraints
- this is consistent with the theory of *extended formulations*

Where we are

- not only are cutting planes (on the original) compatible with DW reformulation
- DW reformulation enables potentially stronger cutting planes

$\rightarrow$ some creativity may be needed to modify the original/pricing problem

Reminder: We want to solve an *Integer* Program

- original problem:

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

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

So far we only solved Linear Programs

- noone ever: "we solved our integer program by column generation!"

So far we only solved Linear Programs

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

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

- we solved the root node so far

$\Rightarrow$ we need to branch!

Thou shalt not branch on single Master Variables

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

Thou shalt not branch on single Master Variables

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

1. the resulting tree is *unbalanced*:
   $\lambda_j \leq \lfloor \lambda_j^* \rfloor$ forbids almost nothing; $\lambda_j \geq \lceil \lambda_j^* \rceil$ enforces much
2. a down branch $\lambda_j \leq \lfloor \lambda_j^* \rfloor$ can be very hard to respect in the pricing problem:
   how to avoid re-generating λ_j?

Branching on Original Variables

- via DW reformulation we arrived at the integer master problem

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

Branching on Original Variables

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

- there are two options for doing so

$\rightarrow$ imposing the branching constraints in the master or in the pricing

// this is the same as in cutting

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

Branching on Original Variables: In the Master

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

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

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

Branching on Original Variables: In the Master

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

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

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

$$\begin{array}{lrcl} \min & (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \alpha_i x_i - \pi_0 & & \\ \text{s.t.} & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{x} & \in & \mathbb{Z}_+^n \end{array}$$

Branching on Original Variables: In the Pricing

- alternatively, impose the branching constraint in the pricing

$$\begin{array}{lrcl} \min & (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \pi_0 & & \\ \text{s.t.} & D\mathbf{x} & \geq & \mathbf{d} \\ & x_i & \leq & \lfloor x_i^* \rfloor \\ & \mathbf{x} & \in & \mathbb{Z}_+^n \end{array}$$

- in this variant, we *additionally* need to forbid master variables that *contradict* the branching decision:

$\rightarrow$ *remove* all variables λ_j from RMP with $x_{ji} > \lfloor x_i^* \rfloor$

$\rightarrow$ this is implemented by imposing a *local upper bound* $\lambda_j \leq 0$

What Original Variables?

- "what if I have no original problem/variables?"

  // i.e., "I did not perform a DW reformulation, I just started generating columns!"

What Original Variables?

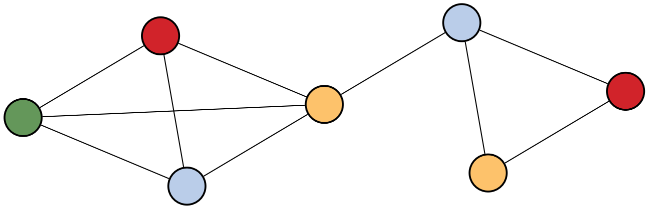
- “what if I have no original problem/variables?”
  // i.e., “I did not perform a DW reformulation, I just started generating columns!”

- you *always* have original variables!

$\longrightarrow$ these are the variables of the pricing problem!

But what Happens when the Master is Aggregated?

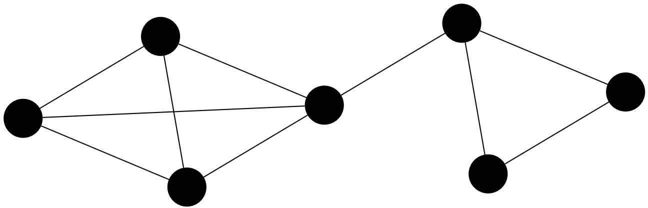
- e.g., our models for vertex coloring
- original variables x_{ic} carry a color



Operations Research | RWTH AACHEN UNIVERSITY

But what Happens when the Master is Aggregated?

- e.g., our models for vertex coloring
- original variables x_{ic} carry a color



- but master variables λ_p represent colorless stable sets

  // neither do pricing problem variables have any color!

We could try a Disaggregation ("Recover the Color")

1. distribute the value λ_p^* of a λ_p variable to the corresponding λ_p^c variables, e.g., evenly

$$\lambda_p = \lambda_p^*$$

$$\lambda_p^1 = \frac{\lambda_p^*}{|C|} \qquad \lambda_p^2 = \frac{\lambda_p^*}{|C|} \qquad \lambda_p^3 = \frac{\lambda_p^*}{|C|} \qquad \cdots \qquad \lambda_p^{|C|} = \frac{\lambda_p^*}{|C|}$$

We could try a Disaggregation ("Recover the Color")

1. distribute the value λ_p^* of a λ_p variable to the corresponding λ_p^c variables, e.g., evenly

$$\lambda_p = \lambda_p^*$$

$$\lambda_p^1 = \frac{\lambda_p^*}{|C|} \qquad \lambda_p^2 = \frac{\lambda_p^*}{|C|} \qquad \lambda_p^3 = \frac{\lambda_p^*}{|C|} \qquad \dots \qquad \lambda_p^{|C|} = \frac{\lambda_p^*}{|C|}$$

2. derive original variable values "as usual"

$$x_{i1} = \sum_{p:i\in p} \lambda_p^1 \quad x_{i2} = \sum_{p:i\in p} \lambda_p^2 \quad x_{i3} = \sum_{p:i\in p} \lambda_p^3 \quad \dots \quad x_{i|C|} = \sum_{p:i\in p} \lambda_p^{|C|}$$

We could try a Disaggregation ("Recover the Color")

1. distribute the value λ_p^* of a λ_p variable to the corresponding λ_p^c variables, e.g., evenly

$$\lambda_p = \lambda_p^*$$

$$\lambda_p^1 = \frac{\lambda_p^*}{|C|} \qquad \lambda_p^2 = \frac{\lambda_p^*}{|C|} \qquad \lambda_p^3 = \frac{\lambda_p^*}{|C|} \qquad \dots \qquad \lambda_p^{|C|} = \frac{\lambda_p^*}{|C|}$$

2. derive original variable values "as usual"

$$x_{i1} = \sum_{p:i\in p} \lambda_p^1 \quad x_{i2} = \sum_{p:i\in p} \lambda_p^2 \quad x_{i3} = \sum_{p:i\in p} \lambda_p^3 \quad \dots \quad x_{i|C|} = \sum_{p:i\in p} \lambda_p^{|C|}$$

- besides the "colorless" pricing problem, one now needs (some) "colorful" ones!

This Always Works

- potential disaggregation, then branching on original variables is a complete branching scheme
- originally proposed by Villeneuve et al. (2005)

- we could do the disaggregation *much* better

$\rightarrow$ Vanderbeck (2011) uses lexicographic disaggregation

- however, drawback always: this (partially) re-introduces the symmetry (in colors)

François is not happy with ~~us~~ the Symmetry

- assume: all generated RMP variables $\lambda_j, j \in J'$ will be finally integer

  // actually, this may be a bit strong, but it never hurts

$\Rightarrow$ for *every* subset $\hat{J} \subseteq J'$ we will have

$$\sum_{j \in \hat{J}} \lambda_j = \beta \in \mathbb{Z}_+ \tag{1}$$

$\Rightarrow$ provide a rule that, should the current master solution be fractional, identifies a subset $\hat{J} \subseteq J$ for which (1) does not hold

- then branch

$$\text{either} \quad \sum_{j \in \hat{J}} \lambda_j \leq \lfloor \beta \rfloor \quad \text{or} \quad \sum_{j \in \hat{J}} \lambda_j \geq \lceil \beta \rceil$$

- Vanderbeck (2000, 2005, 2011) has many wonderful such rules

Popular Special Case: Ryan and Foster (1981)

Ryan and Foster (1981)

Proposition. Let $A \in \{0,1\}^{m \times n}$. For a fractional basic solution $\boldsymbol{\lambda}^* \notin \{0,1\}^n$ to the (LP relaxation of the) *set partitioning problem*

$$\min \left\{ \mathbf{c}^t \boldsymbol{\lambda} \mid A\boldsymbol{\lambda} = \mathbf{1}, \boldsymbol{\lambda} \in \{0,1\}^n \right\}$$

there exist $r, s \in [m]$ with

$$0 < \sum_{j : a_{rj} = a_{sj} = 1} \lambda_j^* < 1 \ .$$

// summing over all subsets $\hat{J}$ that contain both elements, r and s

Ryan-Foster Branching for Set Partitioning

- with this result, the natural branching disjunction is

$$w_{rs} := \sum_{j:a_{rj}=a_{sj}=1} \lambda_j = 0 \qquad \text{or} \qquad w_{rs} = 1$$

- there are different ways of actually doing this, Ryan & Foster (1981) suggest to modify the pricing problem, by adding

$$x_r + x_s \leq 1 \qquad \text{or} \qquad x_r = x_s$$

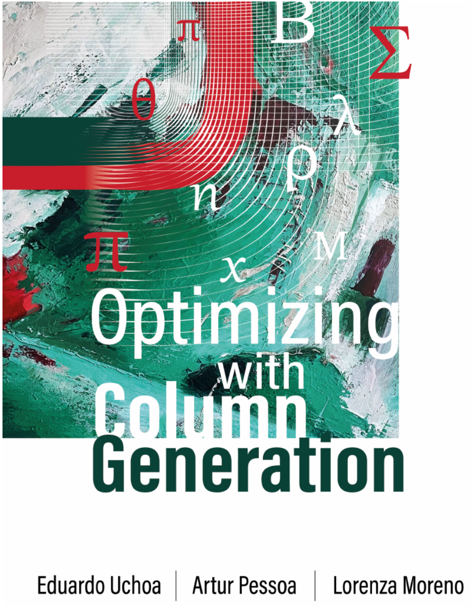
"differ branch" *"same branch"*

// this can be easily handled in some applications

+ eliminate master variables that contradict the branching decision

If you had 15 More Minutes. . .

- Lagrangian relaxation and how it relates to DW reformulation
- Benders decomposition



optimizingwithcolumngeneration.github.io

BRANCH-AND-PRICE

Jacques Desrosiers

Marco Lübbecke

Guy Desaulniers

Jean Bertrand Gauthier

gerad.ca/en/papers/G-2024-36