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

Marco Lübbecke · OR Group · RWTH Aachen University, Germany

@mluebbecke



Prerequisites

→ you already know

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

Goals of this Unit

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

The Cutting Stock Problem

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

The Cutting Stock Problem: Kantorovich (1939, 1960)

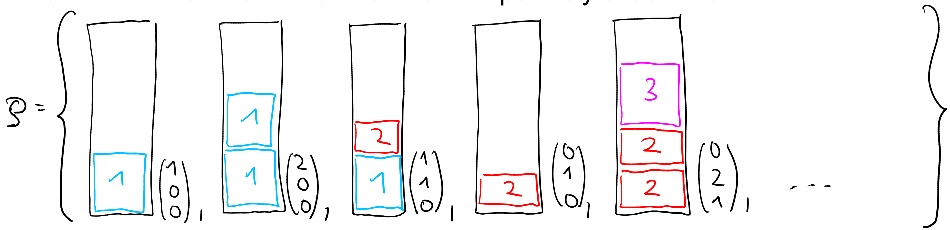
m rolls of length $L \in \mathbb{Z}_+$, n orders of length $\ell_i \in \mathbb{Z}_+$, and demand $d_i \in \mathbb{Z}_+$, $i \in [n] := \{1, \ldots, n\}$

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

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

The Cutting Stock Problem: Gilmore & Gomory (1961)

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



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

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

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

The Cutting Stock Problem: Gilmore & Gomory (1961)

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

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

The Cutting Stock Problem: Gilmore & Gomory (1961)

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

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

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

The Cutting Stock Problem: Gilmore & Gomory (1961)

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

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

- this is an integer program with *maany* variables

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

Why would we care about different Models?

image source: twitter.com, @MurrietaPD

Overview

Column Generation to solve a Linear Program

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

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

- typically, $|J|$ super huge

Column Generation to solve a Linear Program

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

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

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

Column Generation to solve a Linear Program

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

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

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

Column Generation to solve a Linear Program

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

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

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

Column Generation to solve a Linear Program

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

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

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

Naïve Idea for an Algorithm: Explicit Pricing

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

algorithm column generation with explicit pricing

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

repeat
  solve RMP to optimality, obtain $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$;
  compute all $\bar{c}_j = c_j - \boldsymbol{\pi}^t \mathbf{a}_j,\ j \in J$; // computationally prohibitive
  if *there is a variable* λ_{j^*} *with* $\bar{c}_{j^*} < 0$ **then**
    $J' \leftarrow J' \cup \{j^*\}$;
until *all variables* $\lambda_j,\ j \in J$*, have* $\bar{c}_j \geq 0$;

Better Idea: Implicit Pricing

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

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

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

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

The Column Generation Algorithm

algorithm column generation

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

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

Example: Cutting Stock: Restricted Master Problem

- solve LP relaxation of Gilmore & Gomory formulation

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

// one dual variable per order/demand

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

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

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

Example: Cutting Stock: Reduced Cost

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

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

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

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

Example: Cutting Stock: Pricing Problem

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

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

Example: Cutting Stock: Pricing Problem

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

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

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

Example: Cutting Stock: Pricing Problem

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

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

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

- which is a knapsack problem!

Example: Cutting Stock: Pricing Problem

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

1. $z < 0$

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

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

2. $z \geq 0$

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

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

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

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

coefficients a_{ip} obtained from pricing problem solution x_i

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

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

coefficients a_{ip} obtained from pricing problem solution x_i

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

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

coefficients a_{ip} obtained from pricing problem solution x_i

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

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

Why should this work?

Motivation for Column Generation I

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

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

Motivation for Column Generation II

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

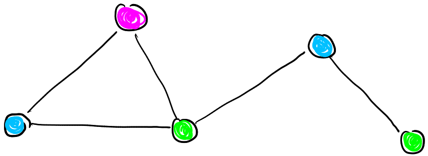
Another Example: Vertex Coloring

Data

$G = (V, E)$ undirected graph

Goal

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



Vertex Coloring: Textbook Model

- notation: C set of available colors

Vertex Coloring: Textbook Model

- notation: C set of available colors

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

Vertex Coloring: Textbook Model

- notation: C set of available colors

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

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

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

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

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

Vertex Coloring: Master Problem

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

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

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

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

Vertex Coloring: Master Problem

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

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

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

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

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

Vertex Coloring: Master Problem

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

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

- 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

Do you know it?

- how does the pricing problem look like?

Vertex Coloring: Pricing Problem

- the pricing problem looks like

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

Vertex Coloring: Pricing Problem

- the pricing problem looks like

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

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

Vertex Coloring: Pricing Problem

- the pricing problem looks like

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

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

- which is a maximum weight independent set problem!