COURSE #14
Thursday, January 18, 2024 5:51 PM

TAKE-HOME EXAM: 5 DAYS

EXAM SUBJECTS: TOMORROW ~NOON
ON CLASSROOM (pdf)

elearning.e-uvt.ro TURNITIN

EVERYONE LOOK FOR THE COURSE UNDER
"BIG DATA"

YOU WILL BE ABLE TO SELF-ENROLL
TO THIS COURSE

SUBMIT HERE LINK
- → a pdf with solns. WORD / LATEX — please no scanned pictures
- → a tar archive with program/programs. python + pulp or julia + JUMP

IMPORTANT HONESTY!

TIMELINE

JAN 19TH NOON EXAM — 5 DAYS → JAN 24TH NOON DEADLINE

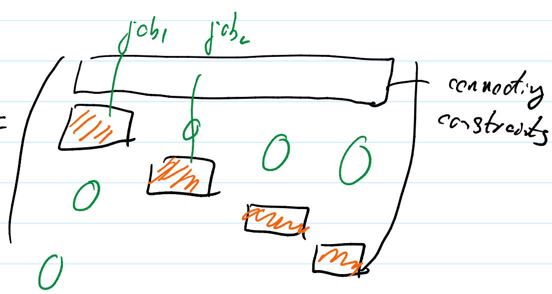
SUBSEQUENT EXAM DATES — in CLASS, ONLINE EXAM

DANTZIG-WOLFE DECOMPOSITION

Method to reformulate large LP problems in a manner that renders them suitable to solving via column generation

$$\begin{cases} \min(c^T x) \\ Ax = b \\ x \geq 0 \end{cases}$$

where $A =$



Example Job scheduling

$$x_{j,m} = \begin{cases} 1 & j \to m \\ 0 & \text{otherwise} \end{cases}$$

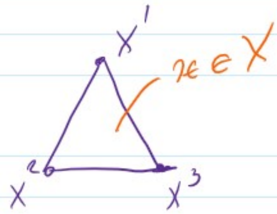
set of variables

$X_{j,m}$

with the same j

$$(1)\quad \begin{cases} \min\ (c^T x) \\ x \in X \\ Ax = b \\ x \geq 0 \end{cases}$$

X → polyhedron generated by the connecting constraints



$$\begin{cases} x = \sum \lambda_i X^i \\ \sum \lambda_i = 1 \\ \lambda_i \geq 0 \end{cases}$$

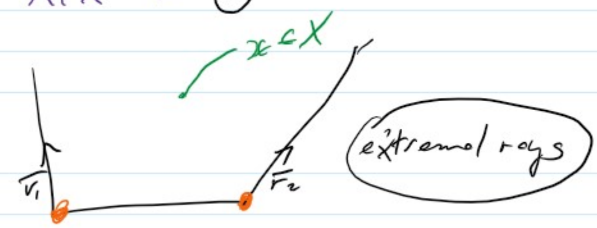
X^i — vertices of X

IDEA MAKE λ_i's the new variables

SUBSTITUTE $x = \sum \lambda_i X^i$ in (1)

PROBLEM

X may be unbounded



THE MINKOWSKI WEIL THM

$$x \in X$$
$$\Updownarrow$$
$$x = \sum_{i=1}^{t} \lambda_i X^i + \sum \mu_j \vec{r_j}$$

X^i — vertices of X; $\vec{r_j}$ — extremal rays

λ_i, μ_j — the new variables

$$\sum_{i=1}^{t} \lambda_i = 1$$

$$\lambda_1 \dots \lambda_t \geq 0$$

$$\mu_j \geq 0$$

DANTZIG WOLFE REFORMULATION → substitution

DANTZIG WOLFE REFORMULATION $\longrightarrow$ substitution

WE'LL NEGLECT EXTREMAL RAYS

$$\left\{\begin{array}{l}\min(c^T x)\\ x \in X\\ Ax = b\\ x \geq 0\end{array}\right. \xRightarrow{x = \sum_{i=1}^{t} \lambda_i X^i} \left\{\begin{array}{l}\min\left(\sum_{i=1}^{t} \underbrace{(c^T X^i)}_{\text{number}} \lambda_i\right)\\ \sum_{i=1}^{t} \underbrace{(A X^i)}_{\text{vectors}} \lambda_i = b\\ \sum_{i=1}^{t} \lambda_i = 1\\ \lambda_i \geq 0\end{array}\right.$$

(P)

$\boxed{z^* = OPT(P)}$

How Do I SOLVE (P) in PRACTICE

X polyhedron $\Rightarrow$ # of vertices X^i may be huge.

$\Downarrow$

# of variables λ_i may be huge!

$\Downarrow$

COLUMN GENERATION

PRACTICAL

- COMMERCIAL SOLVER
- JULIA
  - colana.jl $\Rightarrow$ does D-W for you provided you tell the solver how to decompose the problem.

UNDER DEVELOPMENT TECHNIQUES FOR AUTOMATIC

UNDER DEVELOPMENT

TECHNIQUES FOR AUTOMATIC DECOMPOSITION RESEARCH

YOUTUBE

JUMP-DEV 2020

JUMP-DEV 2019

VIDEOS ABT COLUNA

solves automatically (P) via D-W decomp

| connecting

| B_1

⋮

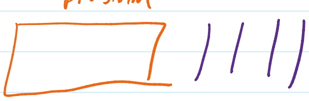
| B_m

anotate constraints

annotate variables } → $B_1, \ldots B_m$.

COLUMN GENERATION

Master problem



solver ⇄ pricing procedure

Start with $\mathcal{I}_0$ of λ_i variables   MASTER PROBLEM

Pricing procedure $\min(\tilde{c}_i) \longrightarrow \tilde{c}_{i^*}$

$\tilde{c}_{i^*} \geq 0 \implies$ STOP

$\tilde{c}_{i^*} < 0 \implies$ Add column i^*

For cutting stock   pricing $\equiv$ knapsack

Pricing   Let $\mathcal{I}$ be the set of columns (vars λ_i)

(initially $\mathcal{I} = \mathcal{I}_0$)

Pricing Let $\mathcal{I}$ be the set of columns (vars λ_i)

(initially $\mathcal{I} = \mathcal{I}_0$)

$$
(P')\begin{cases} \min \left(\sum_{i\in\mathcal{I}} (c_i x^i)\lambda_i \right) \\[2mm] \sum_{i\in\mathcal{I}} (A x^i)\,\lambda_i = b \;\Big|\; y \\[2mm] \sum_{i\in\mathcal{I}} 1 \cdot \lambda_i = 1 \;\Big|\; \alpha \end{cases}
$$

$\boxed{\bar{z} = OPT(P')}$

$\boxed{z^* \leq \bar{z}}$

$\boxed{OPT(D') = (y^*, \bar{\alpha})}$

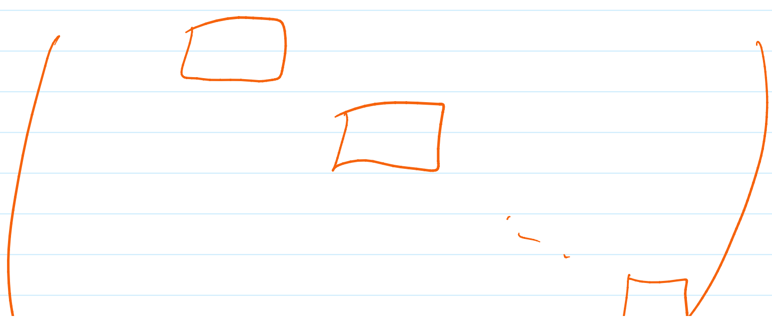
$$
\boxed{\tilde{c}_j = c_j - y^* A x^j - \bar{\alpha} \cdot 1}
$$

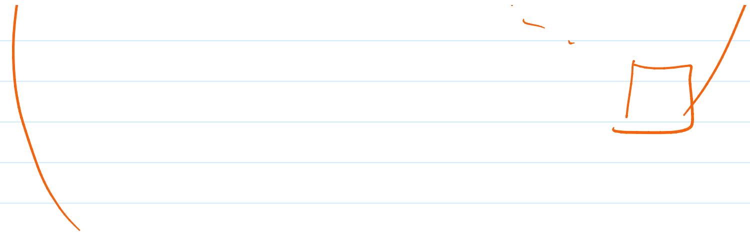
Pricing Problem

$$
(1)\begin{cases} \min \left(c^T x - y^* A x - \bar{\alpha} \right) \\ x \in X \end{cases}
$$

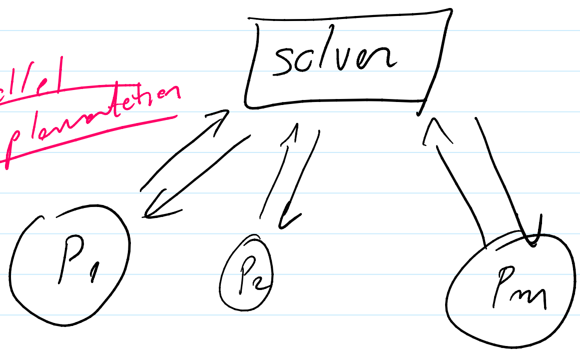
(different if I had extremal rays also)

Pb (1) decompose into r independent problems





parallel implementation



m different pricing procedures

one per block

$\boxed{\text{HOW LONG DO WE RUN? / WHEN DO WE STOP COL. GEN}}$

GOAL $\varepsilon > 0$ WANT $|OPT - SOL| \leq \varepsilon$

Q HOW DO WE FIGURE OUT THAT GOAL IS SATISFIED?

$$(1') \quad \left\{ \max_{x \in X} \left((y^*)^T A x - c^T x + \bar{\alpha} \right) \right.$$

Let $\hat{z}$ be a soln for $(1')$

$$\forall x \in X \quad y^{*T} \underbrace{Ax}_{= b} - c^T x + \bar{\alpha} \leq \hat{z}$$

this is OPTIMUM

$$c^T x \geq \underbrace{(y^*)^T b + \bar{\alpha}}_{\bar{z}} - \hat{z}$$

If $\boxed{OPT(\lambda') \leq \varepsilon}$

$\Downarrow$

$$\bar{z} \geq z^* \geq \bar{z} - \hat{z}$$

$\hat{z} \leq \varepsilon$

<u>Conclusion</u> If $OPT(\text{pricing}) \leq \varepsilon \Longrightarrow$ current soln is at most ε far from optimum.

SOLVE D-W REFORMULATION VIA COL. GEN

(1) Substitution $x \longrightarrow (\lambda_1 \ldots \lambda_t)$

Start with <u>master pb</u> $x \longrightarrow (\lambda_i)_{i \in \mathcal{I}}$

(2) Solve pricing problem

$$\boxed{\tilde{c}_{i^*} = \min(\tilde{c}_i)}$$

$$\boxed{\tilde{c}_{j^*} = \min(\tilde{c}_j)}$$

r independent pricing problems
corresponding to the blocks

j^* is the best soln among the $\le r$
solns suggested by the r pricers

if $\tilde{c}_{j^*} \in [-\varepsilon, 0) \longrightarrow$ STOP

within ε of the optimum

else if $\tilde{c}_{j^*} \geq 0 \longrightarrow$ STOP

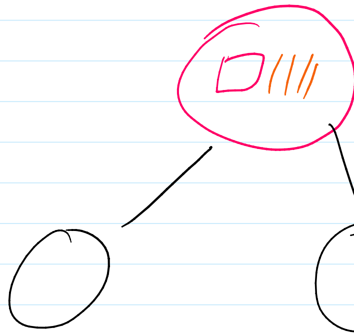
no extra column possible

else
add j^*

CAUTION D-W is for solving LP via column generation.

WHAT IF I WANT TO SOLVE IP ?

Branch - Cut - Price

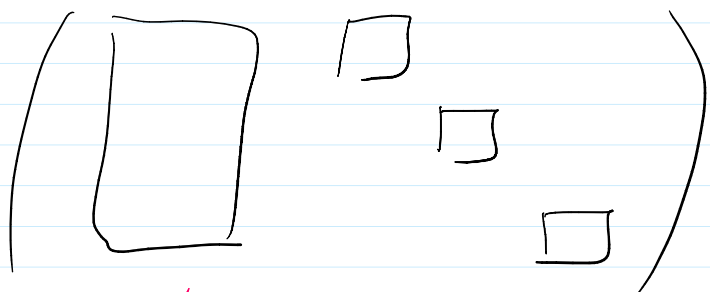


SCIP

WHAT DID I *NOT* TEACH YOU ABOUT

LP/IP?

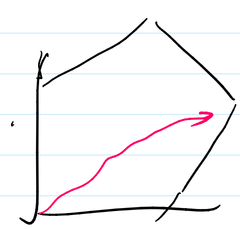
1 BENDERS DECOMPOSITION



connecting variables

2. STABILIZATION — method to speed up col gen in practice.

3. $\boxed{\text{Interior point algorithms}}$



4. sensitivity analysis

5. Lots of Cutting Plane Methods

$\boxed{\text{MATHEMATICAL PROGRAMMING}}$

- quadratic programming
- semidefinite programming
- convex programming
- nonlinear programming

WHERE TO LOOK FOR FURTHER INFO?

INFORMS = professional org. in this area

Coin-OR = software repository

<u>SOLVERS</u> GLPK, CPLEX, GUROBI, SCIP

<u>TEXTBOOKS</u> VASEK CHVÁTAL "Linear programming"

NEMHAUSER & WOLSEY "INTEGER PROGRAMMING"