The Long Road to Practical Decomposition Methods
Part I: Why Leaving the Bed At All?
Part II: A Long Journey Begins

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

AIRO PhD School 2021
&
5th AIRO Young Workshop

“Napoli” — February 9, 2021

Meta–Outline

- Part I: Why Leaving the Bed At All?
- Part II: The Long Journey Begins
- Part III: Many Twists and Turns
- Part IV: A Useful Companion on the Road

Outline – Parts I & II

Part I:
Why Leaving the Bed At All?

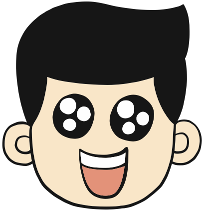
Why Leaving the Bed I:

Why Leaving the Bed I:
“For Science”

Why Leaving the Bed I:
"For Science"
(You Monster)

It All Starts with some Nice Structure

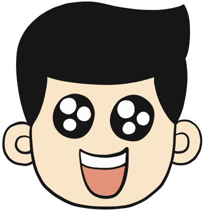
Your favourite structure
HERE



- You (or your boss) have a nice structure you know and love

It All Starts with some Nice Structure

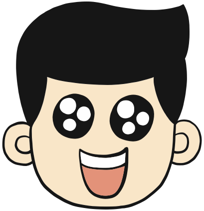
Your favourite structure HERE



- You (or your boss) have a nice structure you know and love
  ... but you start feeling you are scraping the bottom of the barrel
- Want to re-use what you know for solving something else:
  maximise your scientific productivity

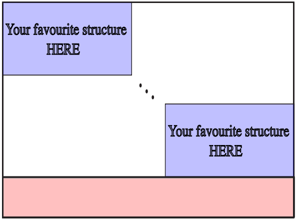
It All Starts with some Nice Structure

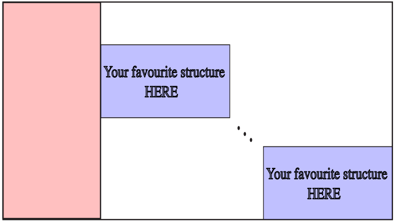
Your favourite structure HERE



- You (or your boss) have a nice structure you know and love
  ... but you start feeling you are scraping the bottom of the barrel
- Want to re-use what you know for solving something else:
  maximise your scientific productivity ("for science", of course)
- Possible in many ways, but two particular ones of interest here

Block-Structured Programs

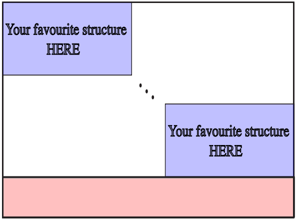




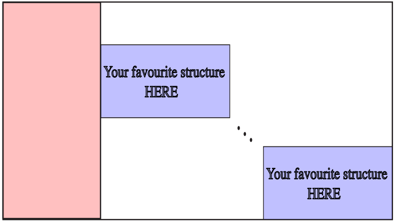
block-diagonal ≡ complicating constraints

staircase-structured ≡ complicating variables

Block-Structured Programs



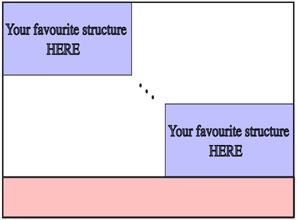
block-diagonal $\equiv$
complicating constraints



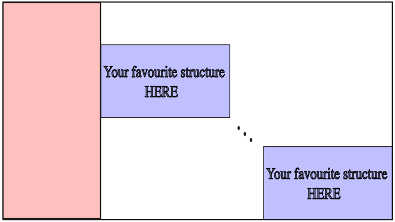
staircase-structured $\equiv$
complicating variables

- Relaxing constraints / fixing variables yields independent subproblems $\Longrightarrow$ much easier because of size and/or structure

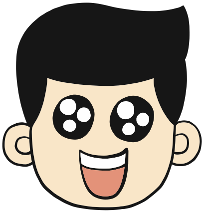
Block-Structured Programs



block-diagonal $\equiv$
complicating constraints

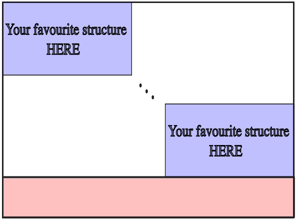


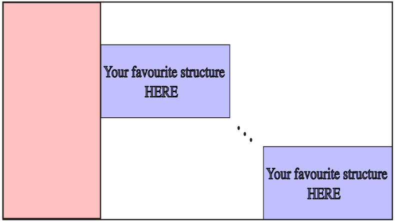
staircase-structured $\equiv$
complicating variables

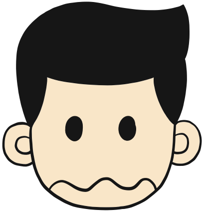


- Relaxing constraints / fixing variables yields independent subproblems $\Longrightarrow$ much easier because of size and/or structure
- Your beloved structure is still there

Block-Structured Programs







block-diagonal $\equiv$ complicating constraints

staircase-structured $\equiv$ complicating variables

- Relaxing constraints / fixing variables yields independent subproblems $\Longrightarrow$ much easier because of size and/or structure
- Your beloved structure is still there
- But you have to understand how to glue back the pieces

My Own Poison: Multicommodity Network Design

- My bosses ♡d shortest paths $\Longrightarrow$ I won multicommodity flows: graph $G = (N, A)$, commodities $K \equiv (s^k\,,\ t^k\,,\ d^k)$ (or general flows)

$$\min \textstyle\sum_{k \in K} \sum_{(i,j) \in A} d^k c^k_{ij} x^k_{ij} + \sum_{(i,j) \in A} f_{ij} z_{ij} \tag{1}$$

$$\sum_{(i,j) \in A} x^k_{ij} - \sum_{(j,i) \in A} x^k_{ji} = \begin{cases} 1 & \text{if } i = s^k \\ 1 & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad i \in N\,,\ k \in K \tag{2}$$

$$\textstyle\sum_{k \in K} d^k x^k_{ij} \leq u_{ij} z_{ij} \qquad (i,j) \in A \tag{3}$$

$$x^k_{ij} \in [0, 1] \qquad (i,j) \in A\,,\ k \in K \tag{4}$$

$$z_{ij} \in \{0, 1\} \qquad (i,j) \in A \tag{5}$$

- Pervasive structure in most of combinatorial optimization
- Many applications: logistic, transportation, telecom, energy, ...
- Multicommodity flows is where actually it all began[1]

[1] Ford, Fulkerson "A Suggested Computation for Maximal Multicommodity Network Flows" *Man. Sci.*, 1958

Does decomposition work?

- Of course it does, in fact with several different approaches:
  - plain[2,3,4] or fancy[5] Lagrangian relaxation, even in parallel[6]
  - structured Dantzig-Wolfe decomposition[7,8]
- Other approaches do as well, though:
  - structured Interior-Point methods[9]
  - structured active-set (simplex) methods[10]
- All in all, lots of fun out of a simple shortest path

[2] F., Gallo "A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min-Cost Flow Problems" *IJoC*, 1999

[3] Crainic, F., Gendron "Bundle-Based Relaxation Methods for Multicommodity [. . .] Network Design" *DAM*, 2001

[4] F., Gendron, Gorgone "On the Computational Efficiency of Subgradient Methods: [. . .]" *Math. Prog. Comput.*, 2017

[5] Grigoriadis, Khachiyan "An Exponential Function Reduction Method for Block-Angular Convex Programs" *Networks*, 1995

[6] Cappanera, F. "Symmetric and Asymmetric Parallelization of a Cost-Decomposition Algorithm [. . .]" *IJoC*, 2003

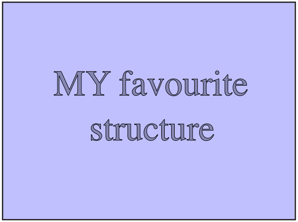
[7] F., Gendron "A Stabilized Structured Dantzig-Wolfe Decomposition Method" *Math. Prog.*, 2013

[8] Mamer, McBride "A Decomposition-Based pricing Procedure for Large-Scale Linear Programs [. . .]" *Man. Sci.*, 2000

[9] Castro "Solving Difficult Multicommodity Problems Through a Specialized Interior-Point Algorithm" *Ann. OR*, 2003

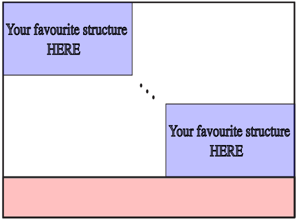
[10] McBride "Progress Made in Solving the Multicommodity Flow Problem" *SIOPT*, 1998

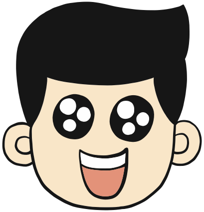
The end result



- My structure

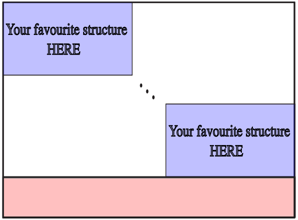
The end result

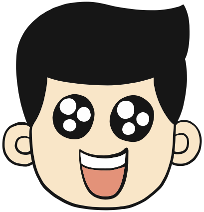




- My structure is decomposition
- Each time there is a valuable structure, I have a new problem to solve
- Give me many structures!

The end result





- My structure is decomposition
- Each time there is a valuable structure, I have a new problem to solve
- Give me many structures!
- Careful what you wish for, you may get it!

Why Leaving the Bed II:

Why Leaving the Bed II: "For Real"

One Day I Got a Phone Call from ...

- ...the Electrical System: mankind's most complex machine
- Many sources of complexity:
  1. the system is just complicated with lots of different machinery inside
  2. electricity is difficult to store $\Longrightarrow$ for the most part it must be produced exactly when needed
  3. electricity is difficult to route, goes where Kirchoff's laws say
  4. growing renewables production is highly uncertain
  5. almost everything is (from slightly to highly) nonlinear
  6. a lot of decisions are combinatorial (on/off)
  7. possibly several actors involved (markets, equilibria, ...)
- All manner of Mixed-Integer NonLinear uncertain optimization problems (or worse) spanning from multi-decades to sub-second
- Let's start "small": the Unit Commitment problem

The Unit Commitment problem

- Schedule a set of generating units over a time horizon (hours/15m in day/week) to satisfy the (forecasted) energy demand at each time
- Gazzillions €€€ / $$$, enormous amount of research[11]
- Different types of production units, many complex constraints:
  - thermal[12] (comprised nuclear): min/max production, min up/down time, ramp rates, start-up cost, modulation, ...
  - hydro[13] (valleys): min/max production, min/max reservoir volume, time delay, pumping, head-dependent energy production, ...
  - non programmable intermittent units: ROR hydro, solar, wind, ...
  - fancy things: small-scale storage, demand response, smart grids, ...
- Plus the electrical network (AC[14]/DC, transmission/distribution) and reliability (primary/secondary reserve, $n-1$ units, ...)

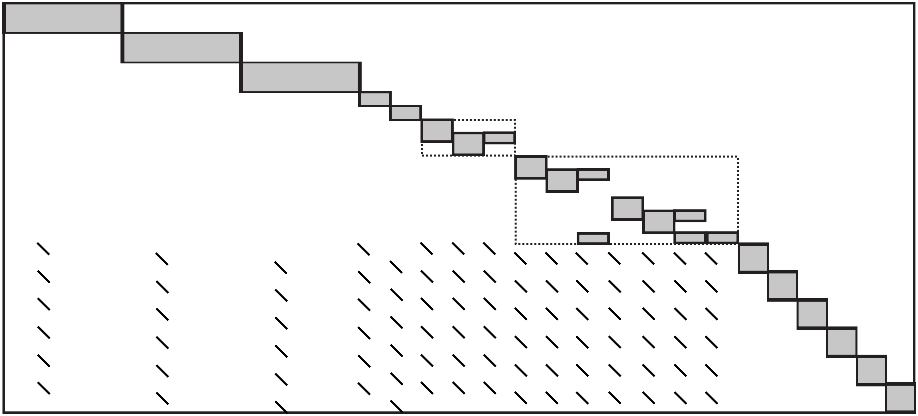
[11] van Ackooij, Danti Lopez, F., Lacalandra, Tahanan "Large-Scale Unit Commitment Under Uncertainty [...]" *AOR*, 2018

[12] F., Gentile "Solving Nonlinear Single-Unit Commitment Problems with Ramping Constraints" *Op. Res.* 2006

[13] van Ackooij, D'Ambrosio, Thomopulos, Trindade "Decomposition and Shortest Path Problem Formulation for Solving the Hydro Unit Commitment and Scheduling in a Hydro Valley" *EJOR*, 2020

[14] Bienstock, Escobar, Gentile, Liberti "Mathematical Programming Formulations for the [AC / OPF]" *4OR*, 2020

All in all

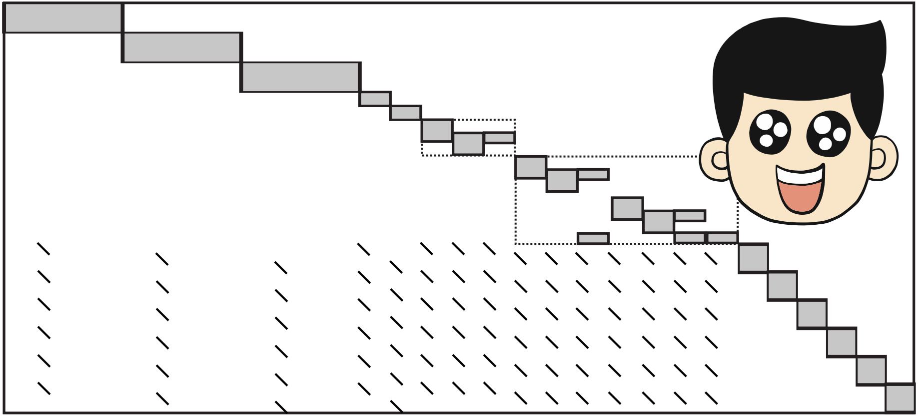


Decomposition methods[15] always been the go-to approach especially in the uncertain case[16]:

[15] Borghetti, F., Lacalandra, Nucci "Lagrangian [...] for Hydrothermal Unit Commitment", *IEEE TPWRS*, 2003

[16] Scuzziato, Finardi, F. "Comparing Spatial and Scenario Decomposition for Stochastic [...]" *IEEE Trans. Sust. En.*, 2018

All in all

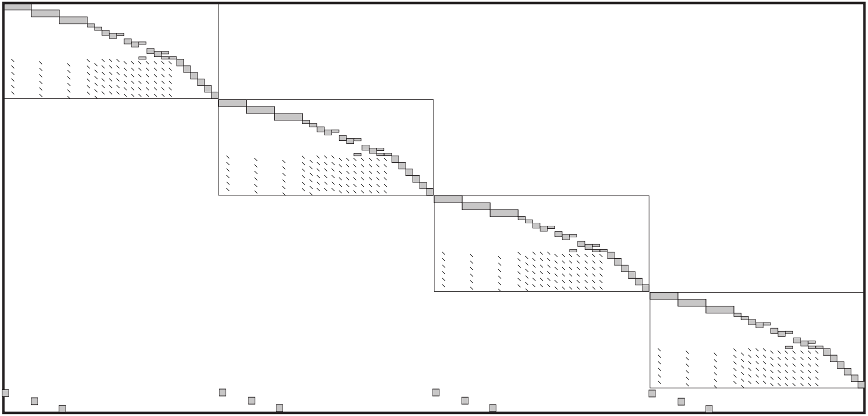


Decomposition methods[15] always been the go-to approach especially in the uncertain case[16]: very good

[15] Borghetti, F., Lacalandra, Nucci "Lagrangian [...] for Hydrothermal Unit Commitment", *IEEE TPWRS*, 2003

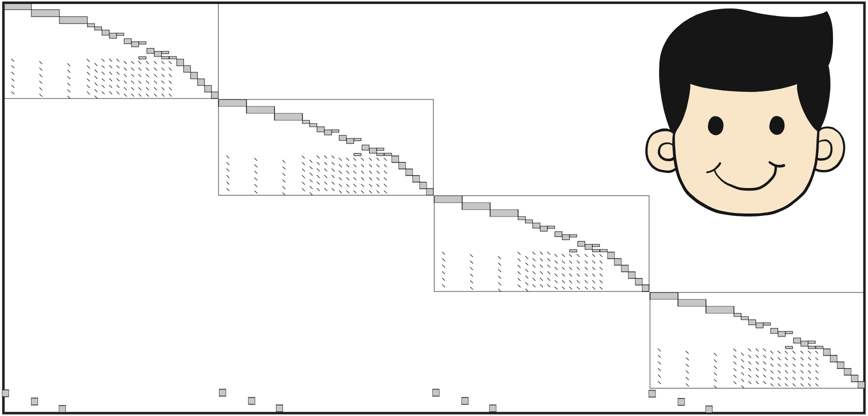
[16] Scuzziato, Finardi, F. "Comparing Spatial and Scenario Decomposition for Stochastic [...]" *IEEE Trans. Sust. En.*, 2018

Then they tell you



. . . that was the operational problem but you must solve the tactical one ≡ that many times over:

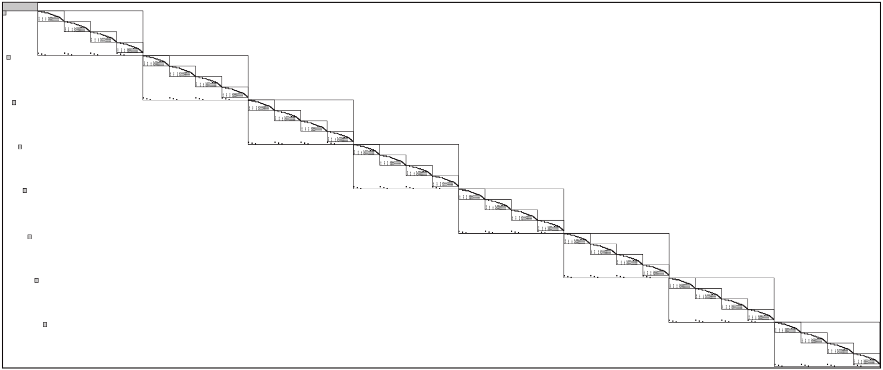
Then they tell you



. . . that was the operational problem but you must solve the tactical one ≡ that many times over:

perhaps still good enough?

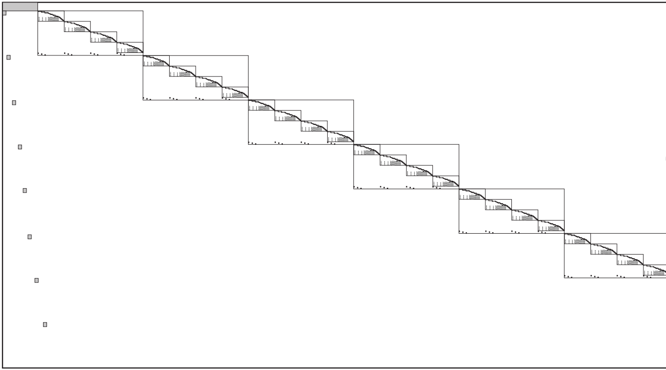
And then it turns out

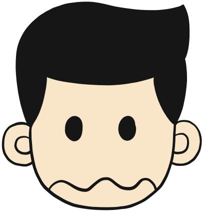


... There's uncertainty and you must do scenarios.

And perhaps use some Stochastic Dual Dynamic Programming to tame it:

And then it turns out

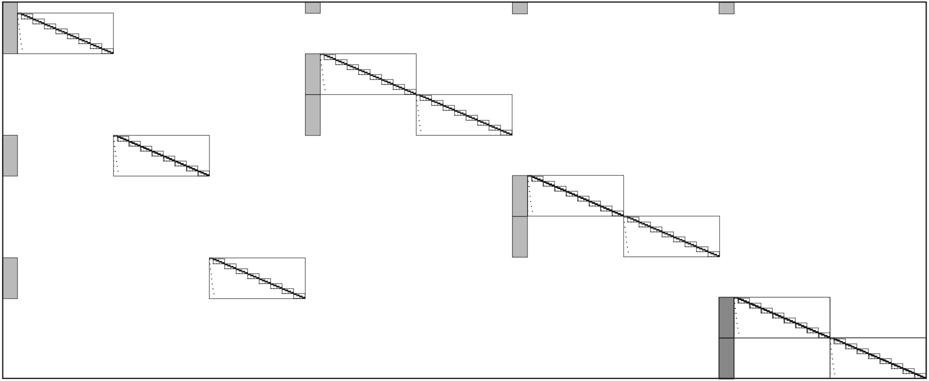




. . . There's uncertainty and you must do scenarios.

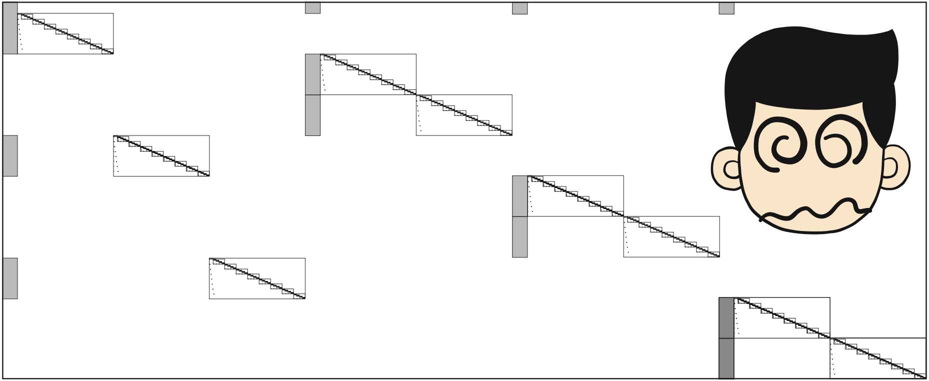
And perhaps use some Stochastic Dual Dynamic Programming to tame it: still feels good?

And finally



Of course what they really wanted to solve is the strategic problem $\equiv$ that many times over again with more scenarios:

And finally



Of course what they really wanted to solve is the strategic problem $\equiv$ that many times over again with more scenarios: I don't feel too good

Too Much of a Good Thing?

- Can you really solve something like this?

Too Much of a Good Thing?

- Can you really solve something like this?
- Surelynot without decomposition
- At least the theory is there (Part II)
- And we can now throw a gazzillion of CPU/GPU cores at it if it helps: yesterday's super[6] is today's smartphone (dishwasher)

Too Much of a Good Thing?

- Can you really solve something like this?
- Surelynot without decomposition
- At least the theory is there (Part II)
- And we can now throw a gazzillion of CPU/GPU cores at it if it helps: yesterday's super[6] is today's smartphone (dishwasher)
- But it won't work by-the-book (Part III)

Too Much of a Good Thing?

- Can you really solve something like this?
- Surelynot without decomposition
- At least the theory is there (Part II)
- And we can now throw a gazzillion of CPU/GPU
  helps: yesterday's super[6] is today's smartphone

- But it won't work by-the-book (Part III)
- And implementing it (in parallel) would be a total nightmare

Too Much of a Good Thing?

- Can you really solve something like this?
- Surelynot without decomposition
- At least the theory is there (Part II)
- And we can now throw a gazzillion of CPU/GPU
  helps: yesterday's super[6] is today's smartphone

- But it won't work by-the-book (Part III)
- And implementing it (in parallel) would be a total nightmare
- Yet, if we can make it we can do tons of other interesting stuff
- That's why we are trying to (Part IV), and you're welcome to join
- That's the plan, let's start from the beginning

Part II:
The Long Journey Begins

Dual decomposition, a.k.a.
Inner Approximation
Dantzig-Wolfe decomposition
Lagrangian Relaxation
Column Generation

Block-diagonal Convex (Linear) Program

- Block-diagonal program: convex X, n "complicating" constraints

$$(\Pi) \qquad \max \{ \, cx \; : \; Ax = b \, , \; x \in X \, \}$$

e.g, $X = \{ \, x \; : \; Ex \leq d \, \} = \bigotimes_{k \in K} \; \left(X^k = \{ \, x^k \; : \; E^k x^k \leq d^k \, \} \right)$
($|K|$ large $\Longrightarrow$ (Π) very large), $Ax = b$ linking constraints

- We can efficiently optimize upon X (much more so than solving the whole of (Π), anyway) for different reasons:
  - a bunch of (many, much) smaller problems instead of a large one
  - X has (the X^k have) structure (shortest path, ...)

- We could efficiently solve (Π) if linking constraints were not there

- But they are (there): how to exploit it?

Dantzig-Wolfe reformulation

- Dantzig-Wolfe reformulation[17]: X convex $\Longrightarrow$ represent it by points

$$X = \{\ x = \sum_{\bar{x} \in X}\ \bar{x}\theta_{\bar{x}}\ :\ \sum_{\bar{x} \in X}\ \theta_{\bar{x}} = 1\ ,\ \ \theta_{\bar{x}} \geq 0 \quad \bar{x} \in X\ \}$$

then reformulate (Π) in terms of the convex multipliers θ

$$(\Pi) \qquad \begin{cases} \max & c\,(\ \sum_{\bar{x} \in X}\ \bar{x}\theta_{\bar{x}}\ ) \\ & A\,(\ \sum_{\bar{x} \in X}\ \bar{x}\theta_{\bar{x}}\ ) = b \\ & \sum_{\bar{x} \in X}\ \theta_{\bar{x}} = 1\ ,\ \ \theta_{\bar{x}} \geq 0 \quad \bar{x} \in X \end{cases}$$

  - only $n+1$ rows, but ∞-ly many columns
  - note that "$\bar{x} \in X$" is an index, not a constraint (θ is the variable)

- A rather semi-infinite program, but "only" $\bar{x} \in ext\, X$ needed

- Not that this makes it any less infinite, unless X is a polytope (compact polyhedron) $\Longrightarrow$ finite set of vertices

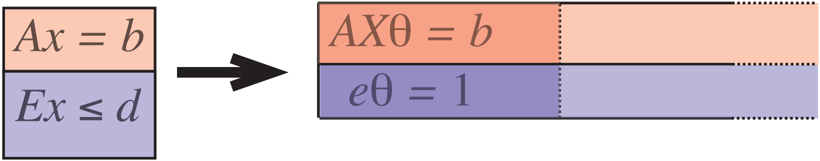
[17] Dantzig, Wolfe "The Decomposition Principle for Linear Programs" *Op. Res.*, 1960

Dantzig-Wolfe reformulation (cont.d)

- Could this ever be a good idea? Actually, it could:
  polyhedra may have few faces and many vertices . . . or vice-versa

n-cube	$\lvert x_i \rvert \leq 1 \quad \forall\, i$	$2n$ faces	2^n vertices
n-co-cube	$\sum_i \lvert x_i \rvert \leq 1$	2^n faces	$2n$ vertices

- Except, most often the number of vertices is too large



  a (linear) program with (exponentially/infinitely) many columns

- But, efficiently optimize over $X \Longrightarrow$ generate vertices ($\equiv$ columns)

Dantzig-Wolfe decomposition $\equiv$ Column Generation

- $\mathcal{B} \subset X$ (small), solve restriction of (Π) with $X \to \mathcal{B}$, i.e.,

$$(\Pi_{\mathcal{B}}) \qquad \begin{cases} \max & \sum_{\bar{x} \in \mathcal{B}} (c\bar{x})\, \theta_{\bar{x}} \\ & \sum_{\bar{x} \in \mathcal{B}} (A\bar{x})\, \theta_{\bar{x}} = b \\ & \sum_{\bar{x} \in \mathcal{B}} \theta_{\bar{x}} = 1 \;,\quad \theta_{\bar{x}} \geq 0 \qquad \bar{x} \in \mathcal{B} \end{cases}$$

  - "master problem" ($\mathcal{B}$ small, not too costly)
  - note how the parentheses have moved: linearity is needed (for now)

- If $\mathcal{B}$ contains the "right" columns, $x^* = \sum_{\bar{x} \in \mathcal{B}} \bar{x}\theta^*_{\bar{x}}$ optimal for (Π)

- How do I tell if $\mathcal{B}$ contains the "right" columns? Use duality

$$(\Delta_{\mathcal{B}}) \qquad \begin{aligned} & \min \left\{ \, yb + v \;:\; v \geq c\bar{x} - y(A\bar{x}) \quad \bar{x} \in \mathcal{B} \, \right\} \\ = \;& \min \left\{ \, f_{\mathcal{B}}(y) = \max \left\{ \, c\bar{x} + y(b - A\bar{x}) \;:\; \bar{x} \in \mathcal{B} \, \right\} \right\} \end{aligned}$$

one constraint for each $\bar{x} \in \mathcal{B}$

Dantzig-Wolfe decomposition $\equiv$ Lagrangian relaxation

- Dual of (Π): $(\Delta) \equiv (\Delta_X)$ (many constraints)
- $f_{\mathcal{B}}$ = lower approximation of Lagrangian function

$$(\Pi_y) \qquad f(y) = \max \{ \, cx + y(b - Ax) \; : \; x \in X \, \}$$

- Assumption: optimizing over X is "easy" for each objective $\Longrightarrow$ obtaining $\bar{x}$ s.t. $f(y) = c\bar{x} + y(b - A\bar{x})$ is "easy"
- Important: (Π_y) Lagrangian relaxation[18], $f(y) \geq v(\Pi) = v(\Delta) \; \forall y$ provided (Π_y) is solved exactly (or at least a $\bar{f} \geq f(y)$ is used)
- Thus, $(\Delta_{\mathcal{B}})$ outer approximation of the Lagrangian Dual

$$(\Delta) \quad \min \{ \, f(y) = \max \{ \, cx + y(b - Ax) \; : \; x \in X \, \} \, \}$$

[18] Geoffrion "Lagrangean Relaxation for Integer Programming" *Math. Prog. Study*, 1974

Lagrangian duality vs. Linear duality

- Note about the LP case ($X = \{ x : Ex \leq d \}$):

$$
\begin{aligned}
(\Delta) \quad & \min \big\{ \, yb + \max \{ \, (c - yA)x \; : \; Ex \leq d \, \} \, \big\} \\
\equiv \quad & \min \big\{ \, yb + \min \{ \, wd \; : \; wE = c - yA \, , \; w \geq 0 \, \} \, \big\} \\
\equiv \quad & \min \big\{ \, yb + wd \; : \; wE + yA = c \, , \; w \geq 0 \, \big\} \\
\equiv \quad & \text{exactly the linear dual of } (\Pi)
\end{aligned}
$$

- y "partial" duals: duals w of $Ex \leq d$ "hidden" in the subproblem

- There is only one duality

- Will repeatedly come in handy

Dantzig-Wolfe decomposition $\equiv$ Dual row generation

- Primal/dual optimal solution $x^*/(v^*, y^*)$ out of $(\Pi_{\mathcal{B}})/(\Delta_{\mathcal{B}})$
- x^* feasible to (Π), so optimal $\iff$ (v^*, y^*) feasible to (Δ)

$$\iff v^* \geq (c - y^*A)x \qquad \forall x \in X$$

$$\iff v^* \geq \max\{\, (c - y^*A)x \;:\; x \in X \,\}$$

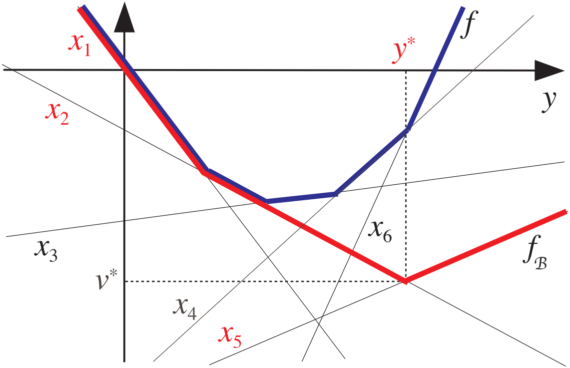
- In fact: $v^* \geq (c - y^*A)\bar{x} \;\equiv\; y^*b + v^* \geq f(y^*) \Longrightarrow$

$$v(\Pi) \geq cx^* = y^*b + v^* \geq f(y^*) \geq v(\Delta) \geq v(\Pi) \Longrightarrow$$

  $x^*/(v^*, y^*)$ optimal

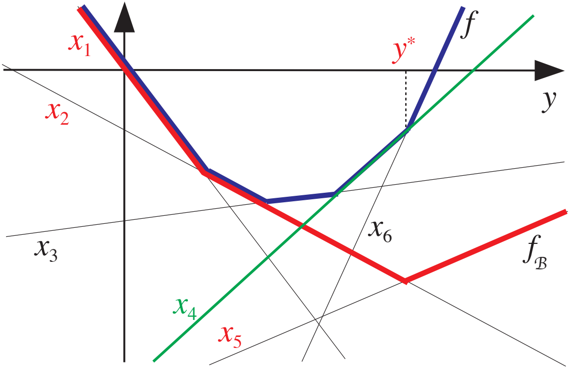
- Otherwise, $\mathcal{B} = \mathcal{B} \cup \{\, \bar{x} \,\}$: add new column to $(\Pi_{\mathcal{B}})$ / row to $(\Delta_{\mathcal{B}})$, rinse & repeat
- Clearly finite if *ext* X is, globally convergent anyway: the Cutting-Plane algorithm for convex programs[19] (applied to (Δ))

[19] Kelley "The Cutting-Plane Method for Solving Convex Programs" *J. of the SIAM*, 1960



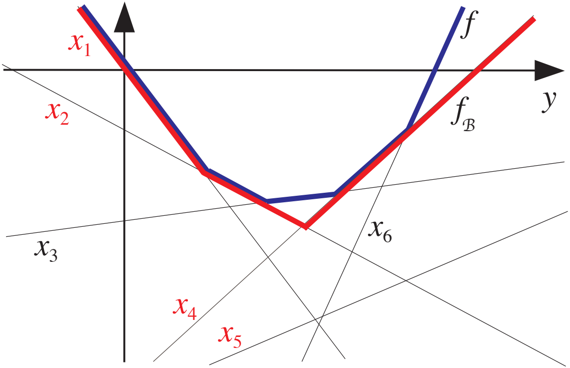
- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$

Geometry of the Lagrangian dual



- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$
- Optimal solution $\bar{x}$ gives separator between (v^*, y^*) and *epi* f

Geometry of the Lagrangian dual



- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$
- Optimal solution $\bar{x}$ gives separator between (v^*, y^*) and *epi* f
- $(c\bar{x}, A\bar{x})$ = new row in $(\Delta_{\mathcal{B}})$ (subgradient of f at y^*)

Dantzig-Wolfe decomposition $\equiv$ Inner Approximation

- "Abstract" view of $(\Pi_{\mathcal{B}})$: $conv(\mathcal{B})$ inner approximation of X

$$(\Pi_{\mathcal{B}}) \qquad \max \{ \; cx \; : \; Ax = b \; , \; x \in conv(\mathcal{B}) \; \}$$

- x^* solves the Lagrangian relaxation of $(\Pi_{\mathcal{B}})$ with y^*, i.e.,

$$x^* \in \operatorname{argmax} \{ \; (c - y^* A)x \; : \; x \in conv(\mathcal{B}) \; \}$$

$$\Longrightarrow (c - y^* A)x \leq (c - y^* A)x^* \text{ for each } x \in conv(\mathcal{B}) \subseteq X$$

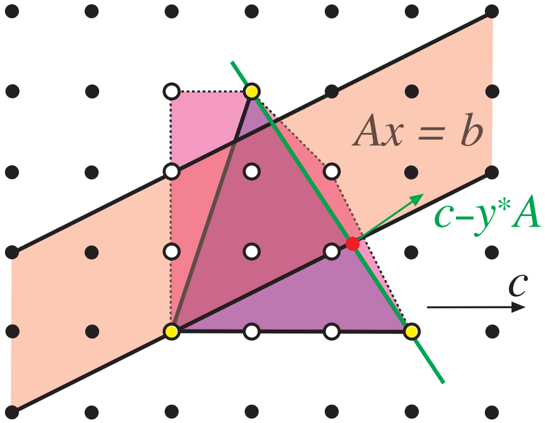
- $(c - y^* A)\bar{x} = \max\{ \; (c - y^* A)x \; : \; x \in X \; \} \geq (c - y^* A)x^*$

- Column $\bar{x}$ has positive reduced cost

$$(c - y^* A)(\bar{x} - x^*) \; = \; (c - y^* A)\bar{x} - cx^* + y^* b \; = \; (c - y^* A)\bar{x} - v^* > 0$$

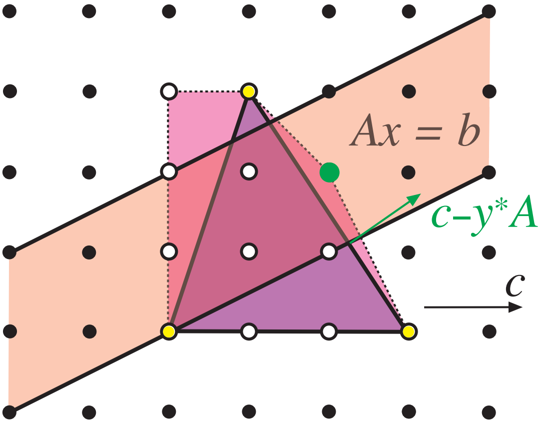
$$\Longrightarrow \bar{x} \notin conv(\mathcal{B}) \Longrightarrow \text{ makes sense to add } \bar{x} \text{ to } \mathcal{B}$$

Geometry of Dantzig-Wolfe/Column Generation



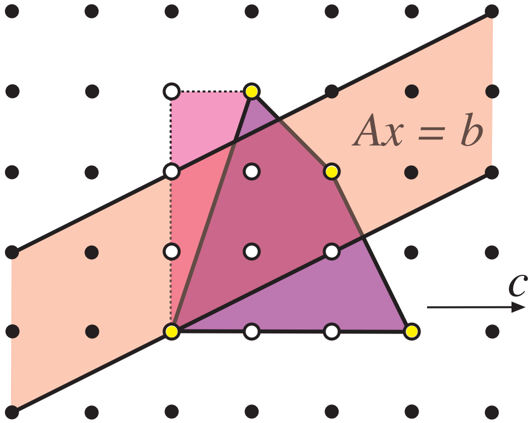
- $c - y^*A$ separates $conv(\mathcal{B}) \cap Ax = b$ from all $x \in X$ better than x^*

Geometry of Dantzig-Wolfe/Column Generation



- $c - y^* A$ separates $conv(\mathcal{B}) \cap Ax = b$ from all $x \in X$ better than x^*
- Thus, optimizing it allows finding new points (if any)

Geometry of Dantzig-Wolfe/Column Generation



- $c - y^*A$ separates $conv(\mathcal{B}) \cap Ax = b$ from all $x \in X$ better than x^*
- Thus, optimizing it allows finding new points (if any)
- Issue: $conv(\mathcal{B}) \cap Ax = b$ must be nonempty

The Unbounded Case

- X unbounded $\iff$ $rec\, X \supset \{0\} \implies f(y) = v(\Pi_y) = \infty$ happens
- $X = conv(\ ext\, X = X_0\ ) + cone(\ ext\ rec\, X = X_\infty\ )$
- $\mathcal{B} = (\,\mathcal{B}_0 \subset X_0\,) \cup (\,\mathcal{B}_\infty \subset X_\infty\,) = \{$ points $\bar{x}$ $\} \cup \{$ rays $\bar{\chi}$ $\} \implies$

$$(\Pi_{\mathcal{B}}) \qquad \begin{cases} \max & c\left(\sum_{\bar{x}\in\mathcal{B}_0} \bar{x}\theta_{\bar{x}} + \sum_{\bar{\chi}\in\mathcal{B}_\infty} \bar{\chi}\theta_{\bar{\chi}}\right) \\ & A\left(\sum_{\bar{x}\in\mathcal{B}_0} \bar{x}\theta_{\bar{x}} + \sum_{\bar{\chi}\in\mathcal{B}_\infty} \bar{\chi}\theta_{\bar{\chi}}\right) = b \\ & \sum_{\bar{x}\in\mathcal{B}_0} \theta_{\bar{x}} = 1 \\ & \theta_{\bar{x}} \geq 0 \quad \bar{x} \in \mathcal{B}_0 \ ,\ \ \theta_{\bar{\chi}} \geq 0 \quad \bar{\chi} \in \mathcal{B}_\infty \end{cases}$$

- In $(\Delta_{\mathcal{B}})$, constraints $y(A\bar{\chi}) \geq c\bar{\chi}$ (a.k.a. "feasibility cuts")
- (Π_{y^*}) unbounded $\iff$ $(c - y^*A)\bar{\chi} > 0$ for some $\bar{\chi} \in rec\, X$ (violated constraint) $\implies \mathcal{B}_\infty = \mathcal{B}_\infty \cup \{\ \bar{\chi}\ \}$
- $(\Delta) = \min\{\ f(y)\ :\ y \in Y\ \}$, (Π_{y^*}) provides either subgradients of f (a.k.a. "optimality cuts"), or violated valid inequalities for Y[19]

Primal decomposition, a.k.a.
Outer Approximation
Benders' decomposition
Resource decomposition

Staircase-structured Convex (Linear) Program

- Staircase-structured program: convex X, "complicating" variables

$$(\Pi) \qquad \max \{\, cx + ez \;:\; Dx + Ez \leq d \;,\; x \in X \,\}$$

e.g, $Dx + Ez \leq d \equiv D_k x + E_k z_k \leq d_k \quad k \in K$ ($|K|$ large) $\Longrightarrow$

$$Z(x) = \{\, z \;:\; Ez \leq d - Dx \,\}$$

$$= \bigotimes_{k \in K} \big(\, Z_k(x) = \{\, z_k \;:\; E_k z_k \leq d_k - D_k x \,\} \big)$$

- We can efficiently optimize upon $Z(x)$ (much more so than solving the whole of (Π), anyway) for different reasons:
  - a bunch of (many, much) smaller problems instead of a large one
  - $Z(x)$ has (the $Z_k(x)$ have) structure (shortest path, ...)
- We could efficiently solve (Π) if linking variables were fixed
- But they are not (fixed): how to exploit it?

Benders' reformulation

- Benders' reformulation: define the concave value function

$$(B) \qquad \max \left\{ \, cx + v(x) = \max\{ \, ez \; : \; Ez \leq d - Dx \, \} \; : \; x \in X \, \right\}$$

(note: clearly $v(x) = -\infty$ may happen)

- Clever trick[20]: use duality to reformulate the inner problem

$$v(x) = \min \left\{ \, w(d - Dx) \; : \; w \in W = \{ \, w \; : \; wE = e \, , \; w \geq 0 \, \} \, \right\}$$

so that W does not depend on x

- As usual, $W = conv(\; ext\, W = W_0 \;) + cone(\; ext\, rec\, W = W_\infty \;) \Longrightarrow$

$$(B) \qquad \begin{array}{lr} \max\, cx + v & \\ v \leq \bar{w}(d - Dx) & \bar{w} \in W_0 \\ 0 \leq \bar{\omega}(d - Dx) & \bar{\omega} \in W_\infty \\ x \in X & \end{array}$$

still very large, but we can generate $\bar{w}$ / $\bar{\omega}$ by computing $v(x)$

[20] Benders "Partitioning Procedures for Solving Mixed-Variables Programming Problems" *Numerische Mathematik*, 1962

Benders' decomposition

- Select (small) $\mathcal{B} = (\, \mathcal{B}_0 \subset W_0 \,) \cup (\, \mathcal{B}_\infty \subset W_\infty \,)$, solve master problem

$$
\begin{array}{lll}
(B_{\mathcal{B}}) & \max \; cx + v & \\
& v \leq \bar{w}(d - Dx) & \bar{w} \in \mathcal{B}_0 \\
& 0 \leq \bar{\omega}(d - Dx) & \bar{\omega} \in \mathcal{B}_\infty \\
& x \in X &
\end{array}
$$

$$
= \max \; \{\; cx + v_{\mathcal{B}}(x) \; : \; x \in X \cap V_{\mathcal{B}} \;\}, \text{ where}
$$

$$
v_{\mathcal{B}}(x) = \min\{\, \bar{w}(d - Dx) \; : \; \bar{w} \in \mathcal{B}_0 \,\} \leq v(x), \; V_{\mathcal{B}} \supseteq dom\, v
$$

- Find (primal) optimal solution x^*, compute $v(x^*)$, get either $\bar{w}$ or $\bar{\omega}$, update either $\mathcal{B}_0$ or $\mathcal{B}_\infty$, rinse & repeat
- Benders' decomposition $\equiv$ Cutting-Plane approach to (B)[19]
- Spookily similar to the Lagrangian dual, ain't it?
- Except, constraints are now attached to dual objects $\bar{w}$ / $\bar{\omega}$

All Are One, One Is All

Benders is Lagrange ...

- Block-diagonal case

$$(\Pi) \quad \max \{ \; cx \; : \; Ax = b \; , \; Ex \leq d \; \}$$

$$(\Delta) \quad \min \left\{ \; yb + wd \; : \; wE + yA = c \; , \; w \geq 0 \; \right\}$$

Think of y as complicating variables in (Δ), you get

$$(\Pi) \quad \max \{ \; cx \; : \; Ax = b \; , \; Ey \leq d \; \}$$

$$(\Delta) \quad \min \left\{ \; yb + \min\{ \; wd \; : \; wE = c - yA \; , \; w \geq 0 \; \} \; \right\}$$

$$= \quad \min \left\{ \; yb + \max\{ \; (c - yA)x \; : \; Ex \leq d \; \} \; \right\}$$

i.e., the Lagrangian dual of (Π)

- The value function of (Δ) is the Lagrangian function of (Π)

...Lagrange is Benders ...

- Dual of (Π) (linear case $X = \{ x : Ax = b \}$)

$$(\Pi) \quad \max \{ cx + ez \ : \ Dx + Ez \leq d \ , \ Ax = b \}$$

$$(\Delta) \quad \min \{ yb + wd \ : \ yA + wD = c \ , \ wE = e \ , \ w \geq 0 \}$$

  Lagrangian dual of the dual constraints $yA + wD = c$ (multiplier x):

$$\begin{aligned}
(\Delta) \ \max \big\{ & \min\{ yb + wd + (c - yA + wD)x \ : \ wE = e \ , \ w \geq 0 \} \big\} \\
= \max \big\{ & cx + \min\{ y(b - Ax) + w(d - Dx) \ : \ wE = e \ , \ w \geq 0 \} \big\} \\
= \max \big\{ & cx + \min\{ y(b - Ax) \} + \\
& \qquad \min\{ w(d - Dx) \ : \ wE = e \ , \ w \geq 0 \} \big\} \\
= \max \big\{ & cx + \max\{ ez \ : \ Dx + Ez \leq e \} \ : \ Ax = b \big\}
\end{aligned}$$

  i.e., Benders' reformulation of (Π)

- The Lagrangian function of (Δ) is the value function of (Π)

...and Both are the Cutting-Plane Algorithm

- Both Lagrange and Benders boil down (changing sign if necessary) to

$$\min \{ \phi(\lambda) \, : \, \lambda \in \Lambda \}$$

  with Λ and ϕ convex, ϕ nondifferentiable

- Both Λ and ϕ only implicitly known via a (costly) oracle: $\bar{\lambda} \longrightarrow$
  - either $\phi(\bar{\lambda}) < \infty$ and $\bar{g} \in \partial\phi(\bar{\lambda}) \equiv \phi(\lambda) \geq \phi(\bar{\lambda}) + \bar{g}(\lambda - \bar{\lambda}) \; \forall \lambda$
  - or $\phi(\bar{\lambda}) = \infty$ and a valid inequality for Λ violated by $\bar{\lambda}$

- "Natural" algorithm: the Cutting-Plane method[19] $\equiv$
  revised simplex method with mechanized pricing in the discrete case

- Natural is not fast, convex nondifferentiable optimization $\Omega(1/\varepsilon^2)$
  and the Cutting-Plane method is much worse than that

- Many variants/other algorithms possible (cf. Part III)

You can apply Lagrange to a Staircase-structured program

- Reformulate a staircase-structured program

$$
\begin{aligned}
\max\ & cx + e'z' + e''z'' \\
& Dx + E'z' \leq d' \ ,\ Dx + E''z'' \leq d'' \\
& x \in X
\end{aligned}
$$

You can apply Lagrange to a Staircase-structured program

- Reformulate a staircase-structured program

$$\max\ cx + e'z' + e''z''$$
$$Dx + E'z' \leq d'\ ,\ \ Dx + E''z'' \leq d''$$
$$x \in X$$

... as a block-diagonal one

$$\max\ c(x' + x'')/2 + e'z' + e''z''$$
$$Dx' + E'z' \leq d'\ ,\ \ x' \in X$$
$$Dx'' + E''z'' \leq d''\ ,\ \ x'' \in X$$
$$x' = x''$$

- Issue: $Dx + Ez \leq d$ must have structure, not $Ez \leq d - Dx$
- Classical approach in stochastic programs[11,16]
  (but beware the multi-stage case)

You can apply Benders' to a Block-diagonal program

- Reformulate a block-diagonal program

$$\begin{aligned}
\max\ & c'x' + c''x'' \\
& E'x' \leq d' \ , \ E''x'' \leq d'' \\
& A'x' + A''x'' = b
\end{aligned}$$

You can apply Benders' to a Block-diagonal program

- Reformulate a block-diagonal program

$$\max\ c'x' + c''x''$$
$$E'x' \leq d'\ ,\ E''x'' \leq d''$$
$$A'x' + A''x'' = b$$

...as a staircase-structured one

$$\max\ c'z' + c''z''$$
$$E'z' \leq d'\ ,\ A'z' = x'$$
$$E''z'' \leq d''\ ,\ A''z'' = x''$$
$$x' + x'' = b$$

- Issue: $Ez \leq d\ ,\ Az = x$ must have structure, not $Ez \leq d$
- Resource decomposition[21] in multicommodity parlance

[21] Kennington, Shalaby "An Effective Subgradient Procedure for Minimal Cost Multicomm. Flow Problems" *Man. Sci.*, 1977

The Nonlinear and Integer Cases

Block-diagonal Convex Nonlinear Programs

- Nonlinear $c(\cdot)$ concave, $A(\cdot)$ component-wise convex, X convex

$$(\Pi) \quad \max \{ \, c(x) \; : \; A(x) \leq b \, , \; x \in X \, \}$$

$$(\Delta) \quad \max \{ \, f(y) = yb + \max \{ \, c(x) - yA(x) \; : \; x \in X \, \} \; : \; y \geq 0 \, \}$$

- Any $\bar{x} \in X$ still gives $f(y) \geq c(\bar{x}) + y(b - A(\bar{x}))$, same $(\Delta_{\mathcal{B}})$ / $(\Pi_{\mathcal{B}})$

- $yA(\bar{x})$ still linear in y even if nonlinear in x

- $c(\sum_{\bar{x} \in \mathcal{B}} \bar{x}\theta_{\bar{x}}) \geq \sum_{\bar{x} \in \mathcal{B}} c(\bar{x})\theta_{\bar{x}}$ ($c(\cdot)$ concave),

  $A(\sum_{\bar{x} \in \mathcal{B}} \bar{x}\theta_{\bar{x}}) \leq \sum_{\bar{x} \in \mathcal{B}} A(\bar{x})\theta_{\bar{x}} \leq b$ ($A(\cdot)$ convex) $\Longrightarrow$

  $(\Pi_{\mathcal{B}})$ safe inner approximation ($v(\Pi_{\mathcal{B}}) \leq v(\Pi)$)

- Basically everything keeps working, but you may need constraint qualification[22] (usually easy to get)

[22] Lemaréchal, Hiriart-Urrity "Convex Analysis and Minimization Algorithms" Springer, 1993

Block-diagonal Nonconvex Nonlinear Programs

- $c(\cdot)$ and/or $A(\cdot)$ and/or X not concave/convex: not much changes

Block-diagonal Nonconvex Nonlinear Programs

- $c(\cdot)$ and/or $A(\cdot)$ and/or X not concave/convex: not much changes except (Π_y) is hard and you are not really solving (Π)
- $yA(\bar{x})$ still linear in y, (Δ) still convex $\equiv$ "convexified" (Π):

  $c(x) = cx \; , \; A(x) = Ax \Longrightarrow (\Delta) \equiv \max \left\{ \, cx \; : \; Ax \leq b \; , \; x \in X^{**} \, \right\}$

  ("$**$" $\equiv$ biconjugate $\equiv$ closed convex envelope / hull)

Block-diagonal Nonconvex Nonlinear Programs

- $c(\cdot)$ and/or $A(\cdot)$ and/or X not concave/convex: not much changes except (Π_y) is hard and you are not really solving (Π)
- $yA(\bar{x})$ still linear in y, (Δ) still convex $\equiv$ "convexified" (Π):

  $c(x) = cx\ ,\ A(x) = Ax \Longrightarrow (\Delta) \equiv \max \{\ cx\ :\ Ax \leq b\ ,\ x \in X^{**}\ \}$

  ("$**$" $\equiv$ biconjugate $\equiv$ closed convex envelope / hull)

  $A(x) = Ax \Longrightarrow (\Delta) \equiv \max \{\ c_X^{**}(x)\ :\ Ax \leq b\ \}$

  ($c_X(\cdot) = c(\cdot) + \imath_X(\cdot)$, $\imath_X \equiv$ indicator function $\equiv 0$ in X, ∞ outside)

  better than $\max \{\ c^{**}(x)\ :\ Ax \leq b\ ,\ x \in X^{**}\ \}$
- General formula ugly to write[23], but better than

  $\max \{\ c^{**}(x)\ :\ A^{**}(x) \leq b\ ,\ x \in X^{**}\ \}$
- "A Lagrangian Dual does not distinguish a set from its convex hull" for better (efficiency) and for worse (not the same problem)

[23] Lemaréchal, Renaud "A Geometric Study of Duality Gaps, with Applications" *Math. Prog.*, 2001

Staircase-structured convex Nonlinear Programs

- $f(x,\cdot)$ and $G(x,\cdot)$ concave, Z convex:

$$(\Pi) \quad \max \{\, f(x,z) \;:\; G(x,z) \geq 0 \,,\; x \in X \,,\; z \in Z \,\}$$

$$(B) \quad \max \;\{\, v(x) \;:\; x \in X \,\}$$

$$\text{where } v(x) = \max\{\, f(x,z) \;:\; G(x,z) \geq 0 \,,\; z \in Z \,\}$$

$(B) \equiv (\Pi)$ without assumptions on $f(\cdot,z)$, $G(\cdot,z)$ and X (hard)

- Which duality would you use? Lagrangian[24], of course

$$v(x) = \min \left\{\, \max\{\, f(x,z) + \lambda G(x,z) \;:\; z \in Z \,\} \;:\; \lambda \geq 0 \,\right\}$$

- Under appropriate constraint qualification, two cases occur:
  - either $\exists\, \bar{\lambda} \geq 0 \,,\; \bar{z} \in Z$ s.t. $v(x^*) = f(x^*,\bar{z}) + \bar{\lambda} G(x^*,\bar{z}) > -\infty$
  - or $v(x^*) = -\infty \Longrightarrow \{\, z \in Z \;:\; G(x^*,z) \geq 0 \,\} = \emptyset \Longrightarrow \exists\, \bar{\nu} \geq 0 \,,\; \bar{z} \in Z$ s.t. $\max\{\, \bar{\nu} G(x^*,z) \;:\; z \in Z \,\} = \bar{\nu} G(x^*,\bar{z}) < 0$

[24] Geoffrion "Generalized Benders Decomposition" *JOTA*, 1972

Staircase-structured convex Nonlinear Programs (cont.d)

- General form of the master problem

$$
(B) \quad \max v \\
\begin{array}{ll}
v \leq \max\{\, f(x,z) + \bar{\lambda}G(x,z) \;:\; z \in Z \,\} & \bar{\lambda} \in \Lambda_0 \\
0 \leq \max\{\, \bar{\nu}G(x,z) \;:\; z \in Z \,\} & \bar{\nu} \in \Lambda_\infty \\
x \in X &
\end{array}
$$

Staircase-structured convex Nonlinear Programs (cont.d)

- General form of the master problem

$$
\begin{array}{lll}
(B) & \max v & \\
& v \leq \max\{\, f(x,z) + \bar{\lambda} G(x,z) \;:\; z \in Z \,\} & \bar{\lambda} \in \Lambda_0 \\
& 0 \leq \max\{\, \bar{\nu} G(x,z) \;:\; z \in Z \,\} & \bar{\nu} \in \Lambda_\infty \\
& x \in X &
\end{array}
$$

- Er . . . how on Earth do you manage those nasty "max"?
- Must be that the "max" can be done independently of x!
- Possible in a few cases, complicated in general

Staircase-structured convex Nonlinear Programs (finish.d)

- Case I, separability: $f(x,z) = f(x) + h(z)$, $G(x,z) = G(x) + H(z)$

$$
\begin{aligned}
(B) \quad & \max f(x) + v \\
& v \leq \bar{\lambda} G(x) + \max\{\; h(z) + \bar{\lambda} H(z) \;:\; z \in Z \;\} \qquad \bar{\lambda} \in \Lambda_0 \\
& 0 \leq \bar{\nu} G(x) + \max\{\; \bar{\nu} G(z) \;:\; z \in Z \;\} \qquad \bar{\nu} \in \Lambda_\infty \\
& x \in X
\end{aligned}
$$

(nonlinear nonconvex cuts, (B) "hard" but it always was so)

- Case II, special forms: $f(z_i)$ concave, univariate

$$
\max \left\{\; \textstyle\sum_i x_i f(z_i) \;:\; \sum_i x_i z_i \leq c \;\;,\;\; z_i \geq 0 \;\;,\;\; Ax \leq b \;\;,\;\; x \geq 0 \right\}
$$

$$
v(x) = \min_\lambda \textstyle\sum_i \max\{\; x_i (f(z_i) - \lambda z_i) \;:\; z_i \geq 0 \;\} + \lambda c
$$

$$
v(x) \leq \textstyle\sum_i x_i \max\{\; (f(z_i) - \bar{\lambda} z_i) \;:\; z_i \geq 0 \;\} + \bar{\lambda} c
$$

can optimize on the z independently from the x $\Longrightarrow$
"normal" linear cuts

Staircase-structured non convex Nonlinear Programs

- $f(x,\cdot)$ and/or $G(x,\cdot)$ not concave and/or Z not convex:

Staircase-structured non convex Nonlinear Programs

- $f(x, \cdot)$ and/or $G(x, \cdot)$ not concave and/or Z not convex: though luck: you basically cannot do anything
- Benders' requires duality, duality requires convexity to work
- Some workarounds possible:
  - Use exact duality for nonconvex problems[25] when available (though!)
  - Approximate the convex hull by some hierarchy[26] (RLT, . . .)
  - Give up duality and use combinatorial Benders' (feasibility) cuts[27]
- In general much harder/less efficient
- Yet, solves the original problem or gives as good a relaxation as the convex approximation of the subproblem is

[25] Guzelsoy, Ralphs "Duality for Mixed-Integer Linear Programs" *ITOR*, 2007

[26] Sen, Sherali "Decomposition [. . .] for Two-Stage Stochastic Mixed-Integer Programming" *Math. Prog.*, 2006

[27] Codato, Fischetti "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming" *Op. Res.*, 2006

Block-diagonal Integer Programs

- Special case: X combinatorial (e.g. , $X = \{ x \in \mathbb{Z}^n \; : \; Ex \leq d \}$)

$$(\Pi) \quad \max \{ \, cx \; : \; Ax = b \, , \; x \in X \, \}$$

$$(\Delta) \quad \min \{ \, yb + \max \{ \, (c - yA)x \; : \; x \in X \, \} \, \}$$

  nothing changes if we can still efficiently optimize over X

  due to size (decomposition) and/or structure (integrality)

- Except we are solving a (potentially good) relaxation of (Π)

$$(\bar{\Pi}) \left\{ \begin{array}{lll} \max & c \left(\sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} \right) & \\ & A \left(\sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} \right) & = b \\ & \sum_{\bar{x} \in X} \theta_{\bar{x}} & = 1 \quad , \quad \theta_{\bar{x}} \geq 0 \quad \bar{x} \in X \end{array} \right.$$

$$\equiv \quad \max \{ \, cx \; : \; Ax = b \, , \; x \in X^{**} = conv(X) \, \}$$

- $\theta_{\bar{x}} \in \mathbb{Z}$ gives a reformulation of (Π); could branch on $\theta_{\bar{x}}$

Block-diagonal Integer Programs (cont.d)

- Good news: $(\bar{\Pi})$ better (not worse) than continuous relaxation ($conv(X) \subseteq \{\ x \in \mathbb{R}^n \ : \ Ex \leq d\ \}$)
- Bad news: (Π_y) "too easy" ($conv(X) = \{\ x \in \mathbb{R}^n \ : \ Ex \leq d\ \}$ $\equiv$ integrality property) $\Longrightarrow$ $(\bar{\Pi})$ same as continuous relaxation
- (Π_y) must be easy, but not too easy (no free lunch)
- Anyway, at best gives good bounds $\Longrightarrow$ Branch & Bound with DW/Lagrangian/CG $\equiv$ Branch & Price
- Although it can be used to drive good heuristics[15,28,29]
- Branching nontrivial: may destroy subproblem structure $\Longrightarrow$ branch on x (but $(\Pi_{\mathcal{B}})$ is on θ)
- Little support from off-the-shelf tools, only SCIP / GCG[30] (for now)

[28] Daniilidis, Lemaréchal "On a Primal-Proximal Heuristic in Discrete Optimization" *Math. Prog.*, 2005
[29] Scuzziato, Finardi, F. "Solving Stochastic [. . .] Unit Commitment with a New Primal Recovery [. . .]" *IJEPES*, 2021
[30] https://scipopt.org, https://gcg.or.rwth-aachen.de

- There may be many choices

$$(\Pi)\quad \max \{\, cx \ :\ Ax = b\,,\ Ex \leq d\,,\ x \in \mathbb{Z}^n \,\}$$

$$(\Pi'_y)\quad \max \{\, cx + y(b - Ax) \ :\ x \in X' = \{\, x \in \mathbb{Z}^n \ :\ Ex \leq d \,\}\,\}$$

$$(\Pi''_w)\quad \max \{\, cx + w(d - Ex) \ : x \in X'' = \{\, x \in \mathbb{Z}^n \ :\ Ax = b \,\}\,\}$$

- The best between (Δ') and (Δ'') depends on integrality of X', X'':
  - if both have it, both (Δ') and $(\Delta'') \equiv$ continuous relaxation
  - if only one has it, the one that does not, but if both don't have it?
- Here comes Lagrangian decomposition[31] (looks familiar?)

$$(\Pi) \equiv \max \{\, (cx' + cx'')/2 \ :\ x' \in X'\,,\ x'' \in X''\,,\ x' = x'' \,\}$$

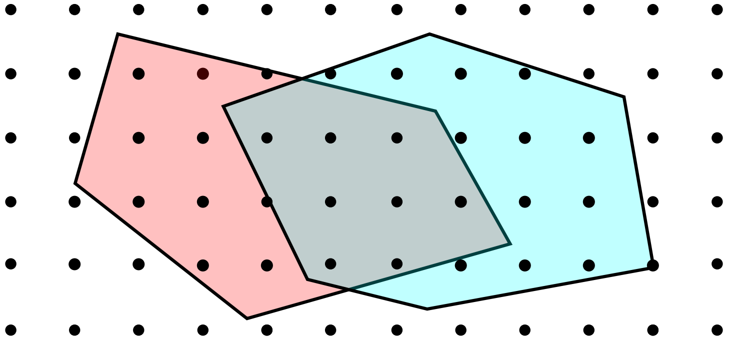
$$(\Pi_\lambda)\quad \max \{\, (c/2 + \lambda)x' \ :\ x' \in X' \,\} + \max \{\, (c/2 - \lambda)x'' \ :\ x'' \in X'' \,\}$$

$$(\bar{\Delta}) \equiv (\bar{\Pi})\quad \max \{\, cx \ :\ x \in conv(X') \cap conv(X'') \,\}$$

- better than both (but need to solve two hard subproblems)

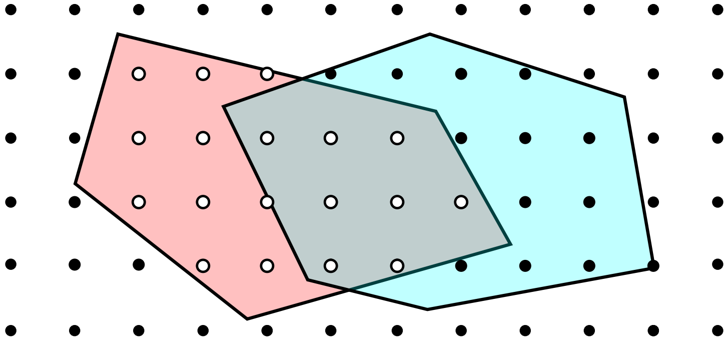
[31] Guignard, Kim "Lagrangean Decomposition: a Model Yielding Stronger Lagrangean Bounds" *Math. Prog.*, 1987

Geometry of Lagrangian Decomposition



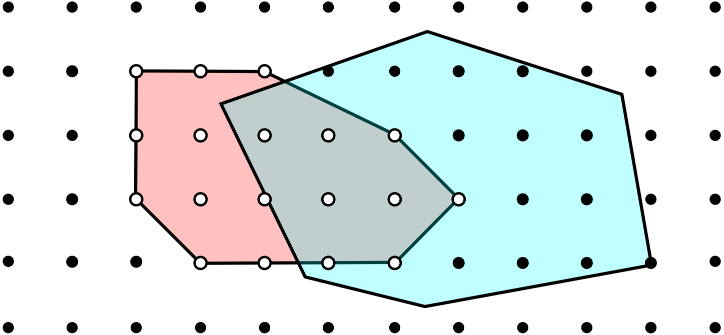
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation

Geometry of Lagrangian Decomposition



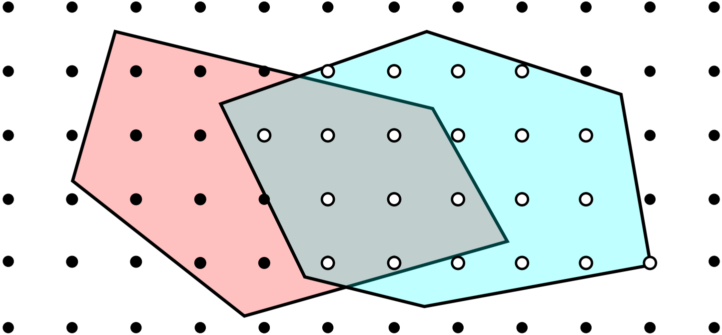
- Intersection between red and blue ≡ grey ≡ continuous relaxation
- Lagrangian relaxation of blue constraints

Geometry of Lagrangian Decomposition



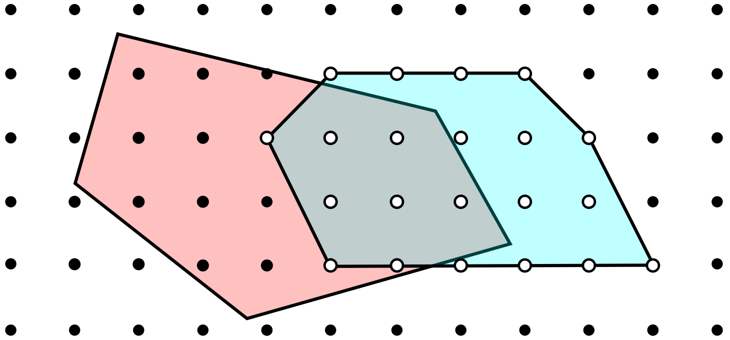
- Intersection between red and blue ≡ grey ≡ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part

Geometry of Lagrangian Decomposition



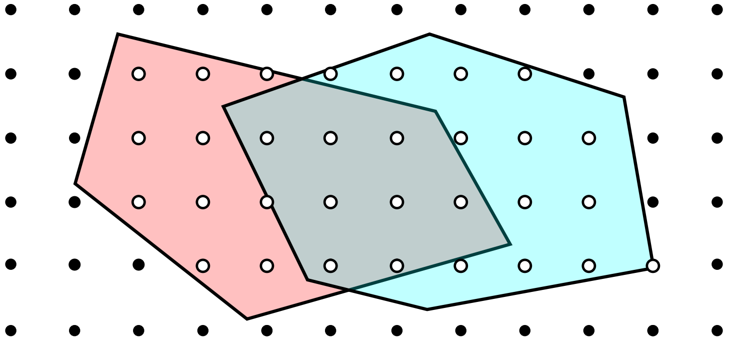
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints

Geometry of Lagrangian Decomposition



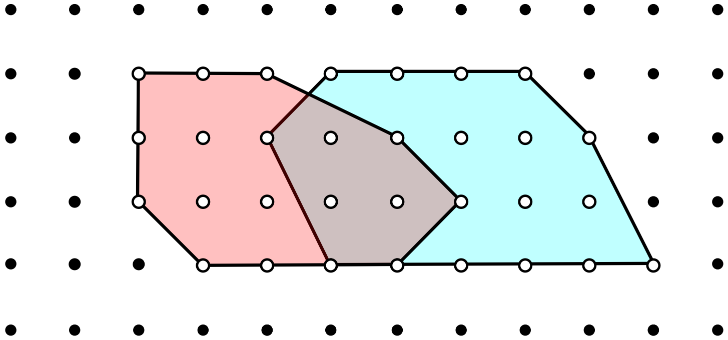
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part

Geometry of Lagrangian Decomposition



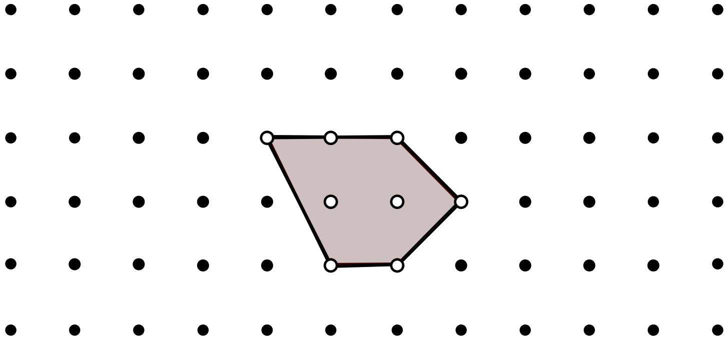
- Intersection between red and blue ≡ grey ≡ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part
- Lagrangian decomposition (both red and blue constraints)

Geometry of Lagrangian Decomposition



- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part
- Lagrangian decomposition (both red and blue constraints) shrinks both $\Longrightarrow$ the grey part more

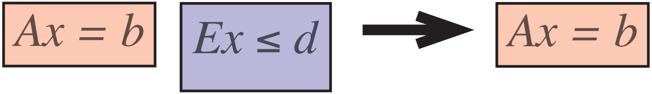
Geometry of Lagrangian Decomposition

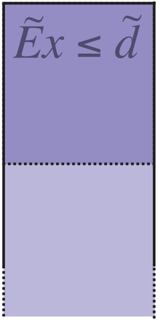


- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part
- Lagrangian decomposition (both red and blue constraints) shrinks both $\Longrightarrow$ the grey part more
- But the intersection of convex hulls is larger (bad) than the convex hull of the intersection

Digression: Alternative Good Formulations for $conv(X)$

- (Under mild assumptions) $conv(X)$ is a polyhedron $\Longrightarrow$ $conv(X) = \{\ x \in \mathbb{R}^n \ :\ \tilde{E}x \leq \tilde{d}\ \}$
- There are good formulations for the problem
- Except, practically all good formulations are too large

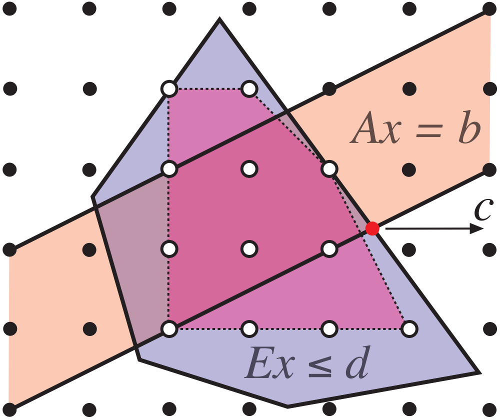




- Very few exceptions (integrality property $\approx$ networks)
- Good part: working in the natural variable space
- But a few more variables do as a lot more constraints

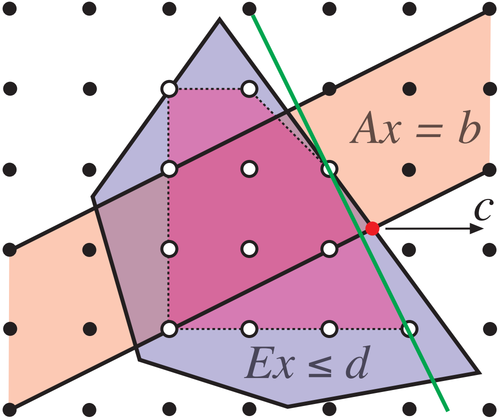
Row generation/polyhedral approaches

- The good news is: rows can be generated incrementally



Row generation/polyhedral approaches

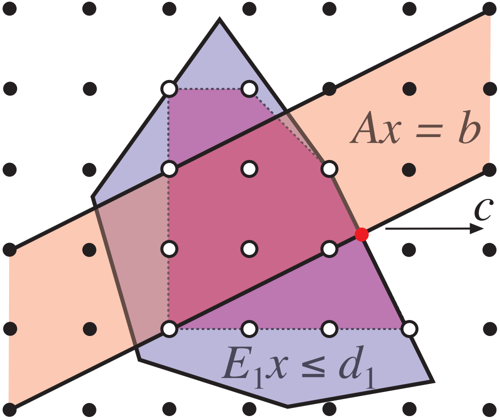
- The good news is: rows can be generated incrementally



- Relevant concept: separator

Row generation/polyhedral approaches

- The good news is: rows can be generated incrementally



- Relevant concept: separator

Branch & Cut

- $\mathcal{R}$ = (small) subset of row(indice)s, $E_{\mathcal{R}} x \leq d_{\mathcal{R}}$ reduced set
- Solve outer approximation to $(\bar{\Pi})$

$$(\bar{\Pi}_{\mathcal{R}}) \qquad \max \{ \, cx \; : \; Ax = b \, , \; E_{\mathcal{R}} x \leq d_{\mathcal{R}} \, \}$$

  feed the separator with primal optimal solution x^*

- Separator for (several sub-families of) facets of $conv(X)$
- Several general approaches, countless specialized ones
- Most often separators are hard combinatorial problems themselves (though using general-purpose MIP solvers is an option[32])
- May tail off, branching useful far before having solved $(\bar{\Pi}_X)$

[32] Fischetti, Lodi, Salvagnin "Just MIP It!" *MATHEURISTICS, Ann. Inf. Syst.*, 2009

Branch & Cut vs. Branch & Price

- Which is best?
- Row generation naturally allows multiple separators
- Very well integrated in general-purpose solvers (but harder to exploit "complex" structures)
- Column generation naturally allows very unstructured separators
- Simpler to exploit "complex" structures (but much less developed software tools)
- Column generation is row generation in the dual
- Then, of course, Branch & Cut & Price (nice, but software issues remain and possibly worsen)

Staircase-structured Integer Programs

- $X = \{ x \in \mathbb{Z}^n \ : \ Ex \leq d \}$ combinatorial:

$$(\Pi) \qquad \max \{ cx + ez \ : \ Ax + Bz \leq b \ , \ x \in X \}$$

  nothing changes . . . except $(B_{\mathcal{B}})$ now is combinatorial $\Longrightarrow$ hard

- However (B_W) now is equivalent to (Π) $\Longrightarrow$ no branching needed unless for solving $(B_{\mathcal{B}})$
- Conversely, everything breaks down if $z \in \mathbb{Z}^m$: there is no (workable[25]) exact dual of an Integer Program
- Can do with "approximated" duals (strong formulations, RLT[26], . . .) but equivalence lost $\Longrightarrow$ branching again
- Alternative route: use Benders' to solve continuous relaxation: Benders' as yet another (strong[33]) cut generator
- Often more efficient and supported by some off-the-shelf solver

[33] Costa, Cordeau, Gendron "Benders, Metric and Cutset Inequalities for Multicommodity [. . .] Network Design" *COAP*, 2009

Conclusions
(for now)

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)
  - where integer/nonconvexity can be (subproblem vs. master problem)

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)
  - where integer/nonconvexity can be (subproblem vs. master problem)
  - where branching/cutting is done (subproblem vs. master problem)

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)
  - where integer/nonconvexity can be (subproblem vs. master problem)
  - where branching/cutting is done (subproblem vs. master problem)
  - where/which nonlinearities can be easily dealt with

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)
  - where integer/nonconvexity can be (subproblem vs. master problem)
  - where branching/cutting is done (subproblem vs. master problem)
  - where/which nonlinearities can be easily dealt with
- But from theory to practice there is a large gulf to be crossed

Conclusions (part I & II)

- General block structure can (and must in some cases) be exploited
- Well-understood main tools: reformulation + duality
- Two different approaches, "primal" and "dual": for linear programs Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting-Plane algorithm[19] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:
  - who is complicating (constraints vs. variables), but tricks ($\equiv$ other reformulations) can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)
  - where integer/nonconvexity can be (subproblem vs. master problem)
  - where branching/cutting is done (subproblem vs. master problem)
  - where/which nonlinearities can be easily dealt with
- But from theory to practice there is a large gulf to be crossed
- Ready your oars, we are going to the sea