Advanced Decomposition Methods
Part I: all is one, one is all

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

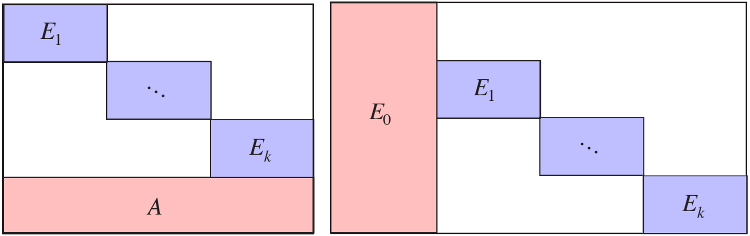
COST/MINO Ph.D. School on Advanced Optimization Methods

Roma — June 8, 2016

Outline

Block-Structured Programs

- Many applications of Mixed-Integer NonLinear Programming are large-scale: millions/billions of variables/constraints
- Good news: (almost) all large-scale problems are block-structured
- Usually several nested forms of structure, but two main ones:



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 (integrality, ...)

Example I: Two-stage Stochastic (Linear) Programs

- Problems involving decisions over time and uncertainty
- First-stage (here-and-now) decisions x, constraints $E_0 x \leq b_0$
- Set S of scenarios, realization known only after deciding x
- Recourse decisions z_s, different for each scenario $s \in S$, constraints $E_0^s x + E_s z_s \leq b_s$
- Minimize here-and-now cost plus average cost of reserve actions
  $$\min \left\{ c_0 x + \textstyle\sum_{s \in S} \pi_s c_s z_s \ : \ E_0 x \leq b_0 \ , \ E_0^s x + E_s z_s \leq b_s \quad s \in S \right\}$$
- Extends to multi-stage (structure repeats "fractally" into each E_s)
- Often other structures inside E, network a common one
- Extends to nonlinear risk measures (CVaR, ...), integer variables, ...
- Many applications: energy[1], water, logistics, telecom, finance, ...

[1] Tahanan, van Ackooij, F., Lacalandra "Large-scale Unit Commitment under uncertainty" *4OR* 2015

Example II: (Linear) Multicommodity Network Design

- Graph $G = (N, A)$, multicommodity network design model

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

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

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

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

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

- $K \equiv$ commodities $\equiv (s^k\,,\ t^k\,,\ d^k)$ (not completely generic)
- Pervasive structure in most of combinatorial optimization
- Many applications: logistic, transportation, telecom, energy, ...
- Nonlinear objective function/constraints (energy, delay[2], ...)

[2] F., Galli, Scutellà "Delay-Constrained Shortest Paths: Approximation Algorithms and SOCP Models" *JOTA*, 2015

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

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

(much more so than solving the whole of (Π), anyway)

- In other words we could efficiently solve (Π) if linking constraints were removed: how to exploit it?

Dantzig-Wolfe reformulation

- Dantzig-Wolfe reformulation[3]: 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

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

  - only $n + 1$ rows (but how 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

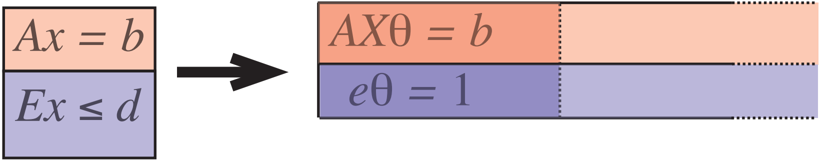
[3] Dantzig, Wolfe "The Decomposition Principle for Linear Programs" *Operations Research* 1960

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.,

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

  - "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

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

  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[4], $f(y) \geq v(\Pi) = v(\Delta)\ \forall y$ 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

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

[4] Geoffrion "Lagrangean relaxation for integer programming" *Mathematical Programming Study* 1974

Lagrangian duality vs. Linear duality

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

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

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

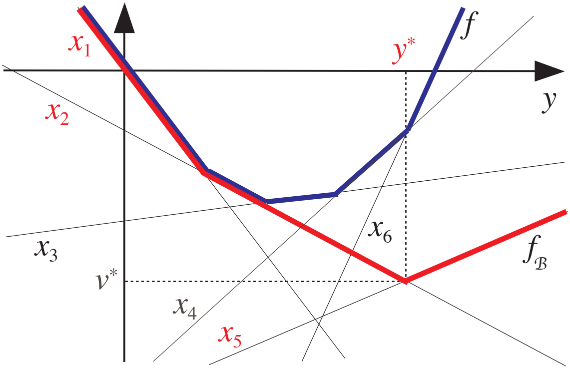
$$v(\Pi) \geq cx^* = y^* b + v^* \geq f(y^*) \geq v(\Delta) \geq v(\Pi) \Longrightarrow$$

  $x^* / (v^*,\ y^*)$ 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 is *ext* X is, globally convergent anyway: the cutting plane algorithm for convex programs[5] (applied to (Δ))

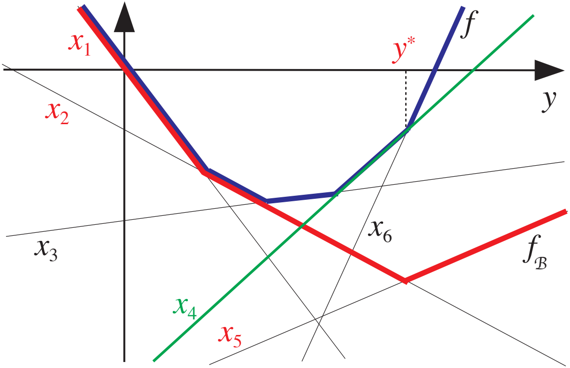
[5] Kelley "The Cutting-Plane Method for Solving Convex Programs" J. of the SIAM, 1960

Geometry of the Lagrangian dual



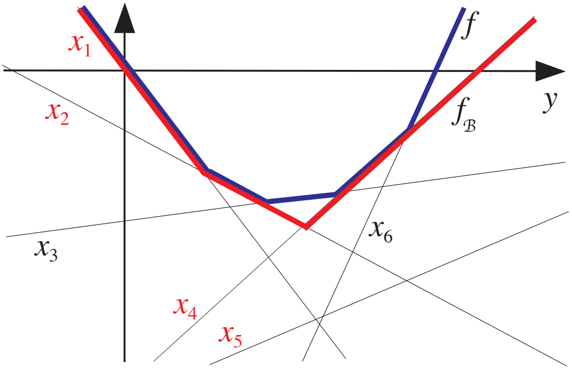
- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$

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.,

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

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

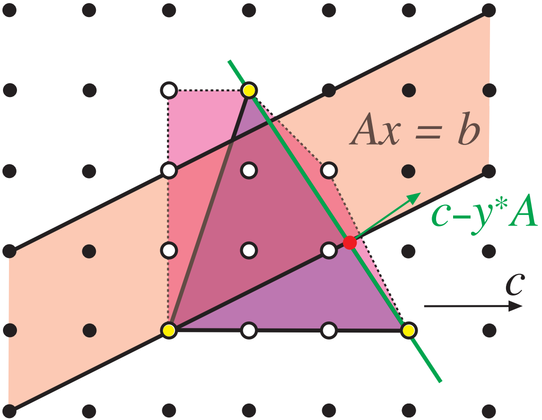
- $(c - y^* A)\bar{x} = \max\{ \, (c - y^* A)x \; : \; x \in X \, \} \geq (c - y^* A)x^*$

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

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

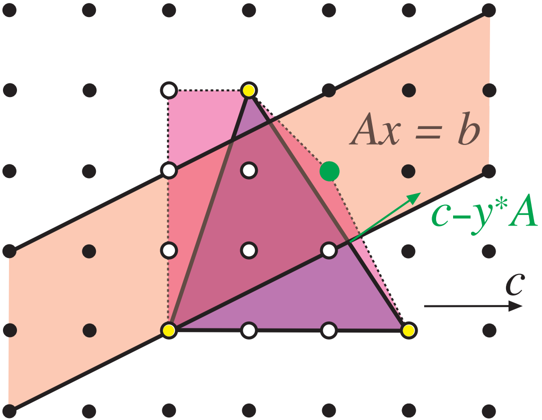
$$\Longrightarrow \bar{x} \notin conv(\mathcal{B}) \Longrightarrow \text{ makes sense to add } \bar{x} \text{ to } \mathcal{B}$$

Geometry of Dantzig-Wolfe/Column Generation



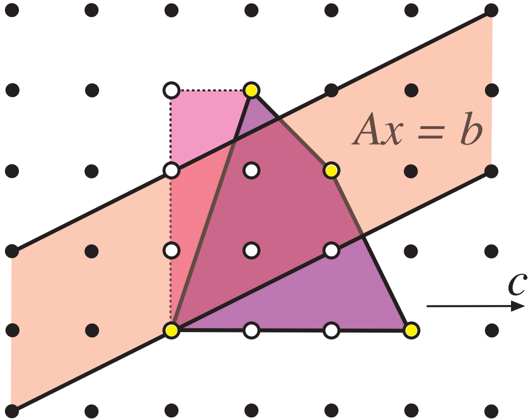
- $c - y^* A$ separates $conv(\mathcal{B}) \cap Ax = b$ from all $x \in X$ better than x^*

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

Extension I: the Unbounded Case

- X unbounded $\iff$ $rec\,X \supset \{0\} \Longrightarrow f(y) = v(\Pi_y) = \infty$ 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 $\bar{\chi}$ $\} \Longrightarrow$

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

- 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) $\Longrightarrow \mathcal{B}_\infty = \mathcal{B}_\infty \cup \{\bar{\chi}\}$
- $(\Delta) = \min\{\ f(y)\ :\ y \in Y\ \}$, (Π_{y^*}) provides either subgradients of f (a.k.a. “optimality cuts”), or violated valid inequalities for Y[5]

Extension II: the Nonlinear Case

- Nonlinear case: $c(\cdot)$ concave, $A(\cdot)$ component-wise 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 $(\Delta_{\mathcal{B}})$ / $(\Pi_{\mathcal{B}})$

- $c(\sum_{\bar{x} \in \mathcal{B}} \bar{x}\theta_{\bar{x}}) \geq \sum_{\bar{x} \in \mathcal{B}} c(\bar{x})\theta_{\bar{x}}$ ($c(\cdot)$ 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[6] (usually easy to get)

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

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, $Dx + Ez \leq d \equiv D_k x + E_k z_k \leq d_k \quad k \in K \; (|K| \text{ large}) \Longrightarrow$

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

- We can efficiently optimize upon $Z(x)$, 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, knapsack, ...)

(much more so than solving the whole of (Π), anyway)

- In other words we could efficiently solve (Π) if linking variables were fixed: how to exploit it?

Benders' reformulation

- Benders' reformulation: define the convex value function

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

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

- Clever trick[7]: 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, $W = conv(\; ext\, W = W_0 \;) + cone(\; ext\, rec\, W = W_\infty \;) \Longrightarrow$

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

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

[7] 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

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

$= \max\ \{\ cx + v_{\mathcal{B}}(x)\ :\ x \in X \cap V_{\mathcal{B}}\ \}$, where
$v_{\mathcal{B}}(x) = \min\{\ \bar{w}(d - Dx)\ :\ \bar{w} \in \mathcal{B}_0\ \} \geq v(x)$, $V_{\mathcal{B}} \supseteq dom\, v$

- 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)[5]
- Spookily similar to the Lagrangian dual, ain't it?
- Except, constraints are now attached to dual objects $\bar{w}$ / $\bar{\omega}$

Benders is Lagrange ...

- Block-diagonal case

$$(\Pi) \quad \max \{\, cx \;:\; Ax = b \,,\; Ex \le d \,\}$$
$$(\Delta) \quad \min \left\{\, yb + wd \;:\; wE + yA = c \,,\; w \ge 0 \,\right\}$$

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

$$(\Pi) \quad \max \{\, cx \;:\; Ax = b \,,\; Ey \le d \,\}$$
$$(\Delta) \quad \min \left\{\, yb + \min\{\, wd \;:\; wE = c - yA \,,\; w \ge 0 \,\} \,\right\}$$
$$= \min \left\{\, yb + \max\{\, (c - yA)x \;:\; Ex \le d \,\} \,\right\}$$

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):

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

  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 to

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

  with Λ and ϕ convex, nondifferentiable, both only implicitly known by means of a (potentially costly) oracle that, given $\bar{\lambda}$, provides:
  - either $\phi(\bar{\lambda}) < \infty$ and $\bar{g} \in \partial\phi(\bar{\lambda}) \equiv \phi(\lambda) \geq \phi(\bar{\lambda}) + \bar{g}(\lambda - \bar{\lambda})$
  - or $\phi(\bar{\lambda}) = \infty$ and a valid cut for Λ violated by $\bar{\lambda}$

- "Natural" algorithm: the Cutting Plane method[5] $\equiv$ revised simplex method with mechanized pricing in the discrete case

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

The Nonlinear Case

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

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

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

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

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

- Which duality would you use? Lagrangian[8], 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. $\max\{\, \bar{\nu} G(x^*, z) \;:\; z \in Z \,\} = \bar{\nu} G(x^*, \bar{z}) < 0$

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

- 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!
- Example: $f(z_i)$ concave, univariate

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

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

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

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

The Integer Case

Block-structured Integer Programs

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

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

- The Lagrangian dual is

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

nothing changes if we can still efficiently optimize over X, e.g. due to size (decomposition) and/or structure (integrality)

- ...except we are solving a different problem:

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

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

i.e., a (potentially good) relaxation of (Π)

Block-structured 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: if (Π_y) "too easy" $(conv(X) = \{\, x \in \mathbb{R}^n \;:\; Ex \leq d \,\}$, a.k.a. integrality property), then $(\bar{\Pi})$ same as continuous relaxation
- Trade-off: (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
- Branching nontrivial: may destroy subproblem structure $\Longrightarrow$ branch on x (but $(\Pi_{\mathcal{B}})$ is on θ)
- Lamentably little support from off-the-shelf tools: master problem gives a valid bound only at termination, although subproblem always gives one (but not associated to continuous feasible solution)

Digression: How to Choose your Lagrangian relaxation

- 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[9] (scale by $1/2$)

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

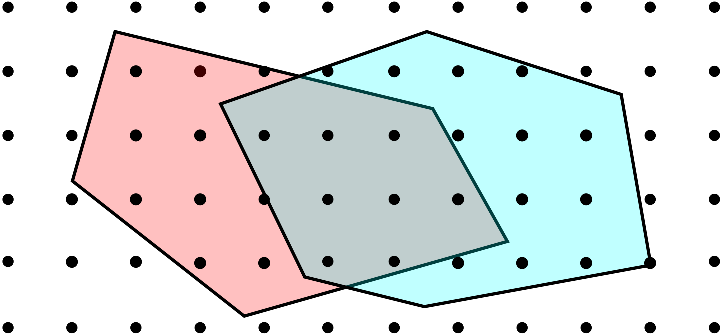
$$(\Pi_\lambda) \qquad \max \{ \, (c + \lambda)x' \; : \; x' \in X' \, \} + \max \{ \, (c - \lambda)x'' \; : \; x'' \in X'' \, \}$$

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

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

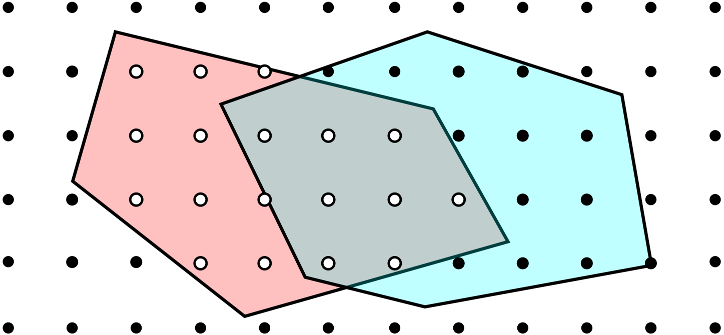
[9] Guignard, Kim "Lagrangean decomposition: a model yielding stronger lagrangean bounds" *Math. Prog.*, 1987

Geometry of Lagrangian Decomposition



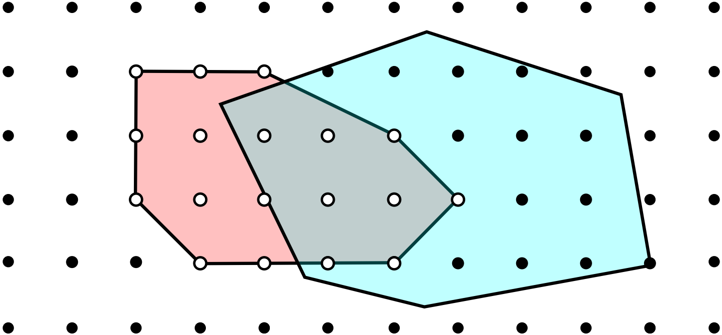
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation

Geometry of Lagrangian Decomposition



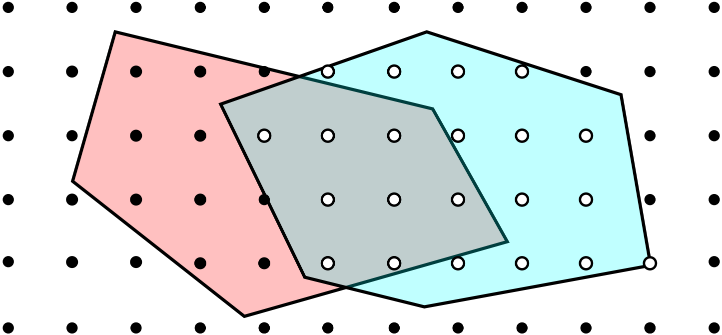
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints

Geometry of Lagrangian Decomposition



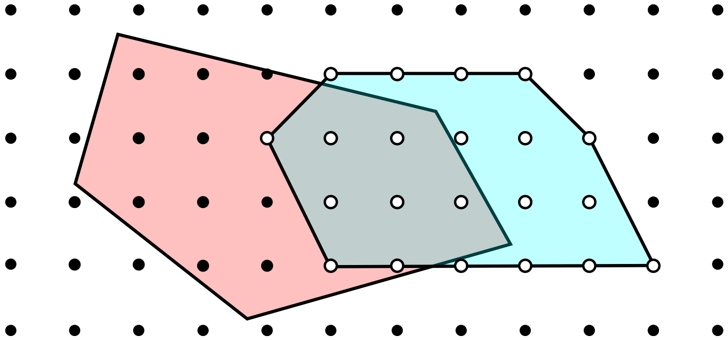
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part

Geometry of Lagrangian Decomposition



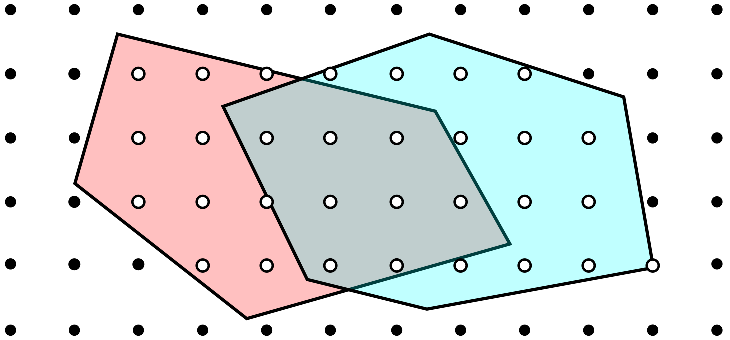
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints

Geometry of Lagrangian Decomposition



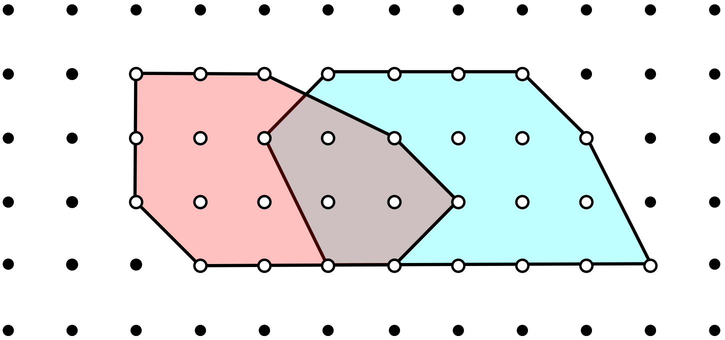
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part

Geometry of Lagrangian Decomposition



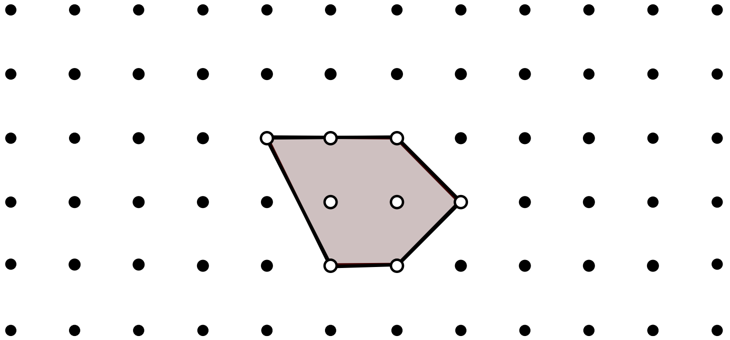
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part
- Lagrangian decomposition (both red and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

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



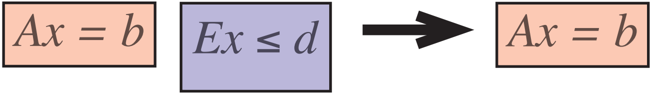
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part
- Lagrangian decomposition (both red and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

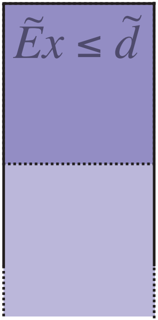
Digression: Alternative Good Formulations for $conv(X)$

- (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





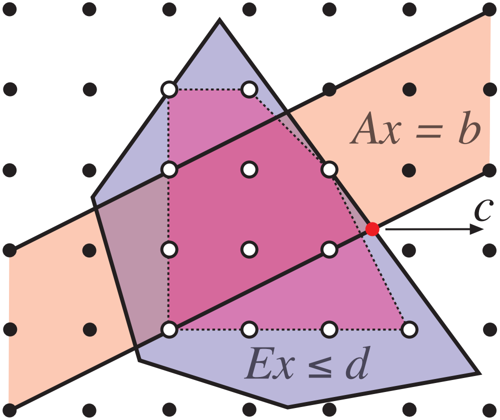
- Very few exceptions (integrality property $\approx$ networks)

- Good part: working in the natural variable space

- But a few more variables do as a lot more constraints

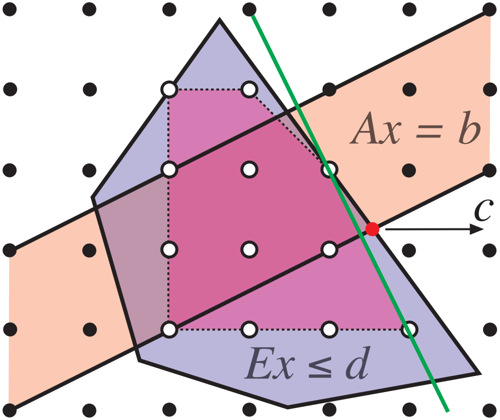
Row generation/polyhedral approaches

- The good news is: rows can be generated incrementally



Row generation/polyhedral approaches

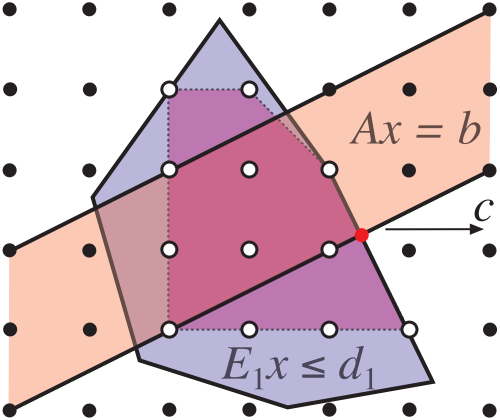
- The good news is: rows can be generated incrementally



- 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 codes is an option[10])
- May tail off, branching useful far before having solved $(\bar{\Pi}_X)$

[10] 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

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

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

- Nothing changes ... except $(B_{\mathcal{B}})$ now is combinatorial $\Longrightarrow$ hard
- However (B_W) now is equivalent to $(\bar{\Pi})$ $\Longrightarrow$ no branching needed (unless for solving $(B_{\mathcal{B}})$) $\Longrightarrow$ no Branch & Benders'
- Conversely, everything breaks down if $z \in \mathbb{Z}^m$: there is no (workable, exact) dual of an Integer Program
- Can do with "approximated" duals (strong formulations, RLT[11], ...) but equivalence lost $\Longrightarrow$ branching again

[11] Sen, Sherali "Decomposition with branch-and-cut approaches for two-stage stochastic MIP" *Math. Prog.*, 2006

Example Applications, a.k.a. the Reformulation before the Reformulation

(Very) Classical decomposition approaches for (2SILP)

- Here-and-now decisions are naturally complicating variables
- The (expected) value function decomposes by scenario

$$v(x) = c_0 x + \sum_{s \in S} \pi_s \min \{\, c_s z_s \,:\, E_s z_s \leq b_s - E_0^s x \,\}$$

- Alternative approach: split variables, introduce copy constraints

$$\begin{aligned} \min\ & c_0 x + \textstyle\sum_{s \in S} \pi_s c_s z_s \\ & E_0 x \leq b_0 \\ & E_0^s x_s + E_s z_s \leq b_s \ ,\ \ x_s = x \qquad\qquad s \in S \end{aligned}$$

  relax them in a Lagrangian fashion

- Lagrangian approach chooses all variables for all scenarios (no unfeasibility), tries to make here-and-now agree by changing prices
- Difference more pronounced in multi-stage programs

Classical decomposition approaches for (MCND)

- Design (z) variables are "naturally" linking / complicating
  - What remains is flow/paths: convex even if integer
  - Benders' cuts are metric inequalities[12] defining the multiflow feasibility
- Resource decomposition[13]: add artificial linking variables

$$d^k x_{ij}^k \leq u_{ij}^k \qquad , \qquad \sum_{k \in K} u_{ij}^k \leq u_{ij}$$

- Different possible linking constraints:
  - (3): $\Longrightarrow$ flow (shortest path) relaxation (integrality property $\equiv$ "easy")
  - (2): $\Longrightarrow$ knapsack relaxation (only one integer variable per problem)
  - different efficiency (algorithm-dependent[14,15]), others possible

---

[12] Costa, Cordeau, Gendron "Benders, metric and cutset inequalities for multicommodity capacitated network design" *Comput. Optim. Appl.*, 2009

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

[14] Crainic, F., Gendron "Bundle-based relaxation methods for multicommodity capacitated fixed charge network design" *Disc. Appl. Math.* 2001

[15] F., Gendron, Gorgone "On the computational efficiency of subgradient methods: a case study in combinatorial optimization" *Math. Prog. Comp.* (submitted), 2015

Conclusions
(for now)

Conclusions (part I)

- Structured (Integer) Programs are challenging, but structure can be exploited: main tools are reformulation + duality
- Two different approaches, "primal" and "dual"
- 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
- (For linear programs) Lagrange is Benders' in the dual, and vice-versa
- Both boil down to the 50+-years old Cutting Plane algorithm[5]
- Has it aged well? We'll see tomorrow