An introduction to energy optimization in SMS++
Part III: a quick recap of decomposition techniques

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

EdF Labs — May 25-26, 2023

Meta–Outline

- Part I: SMS++ basics & energy-related components
- Part II: hands-on with SMS++ for energy optimization
- Part III: a quick recap of decomposition techniques
- Part IV: decomposition & energy optimization in SMS++

Outline – Part III

1. Dual decomposition (Dantzig-Wolfe/Lagrangian/Column Generation)
2. Primal decomposition (Benders'/Resource)
3. All Are One, One Is All
4. Dual Decomposition: the Nonlinear and Integer Cases
5. Alternative Good Formulations for $conv(X)$
6. Primal Decomposition: the Nonlinear and Integer Cases
7. Decomposition-aware modelling systems
8. Conclusions (for now)

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[1]: X convex $\Longrightarrow$ represent it by points

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

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

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

  - 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

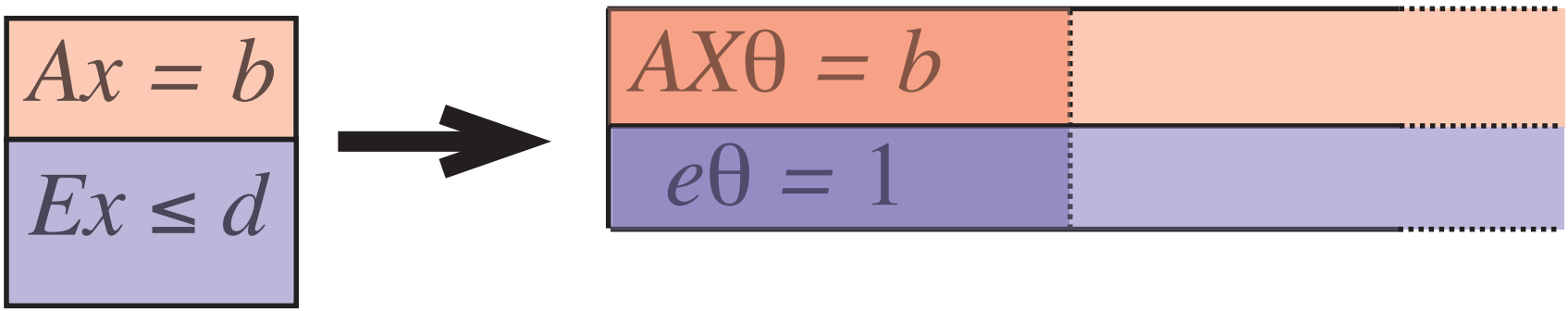
[1] Dantzig, Wolfe "The Decomposition Principle for Linear Programs" *Op. Res.*, 1960

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.,

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

  - "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

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

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

Dantzig-Wolfe decomposition ≡ Lagrangian relaxation

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

$$(\Pi_y) \quad f(y) = \max\{ cx + y(b - Ax) \ : \ x \in X \} \geq f_{\mathcal{B}}(y)$$

- 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[2]

$$f(y) \geq v(\Pi) = v(\Delta) \qquad \forall y$$

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

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

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

Lagrangian duality vs. Linear duality

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

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

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

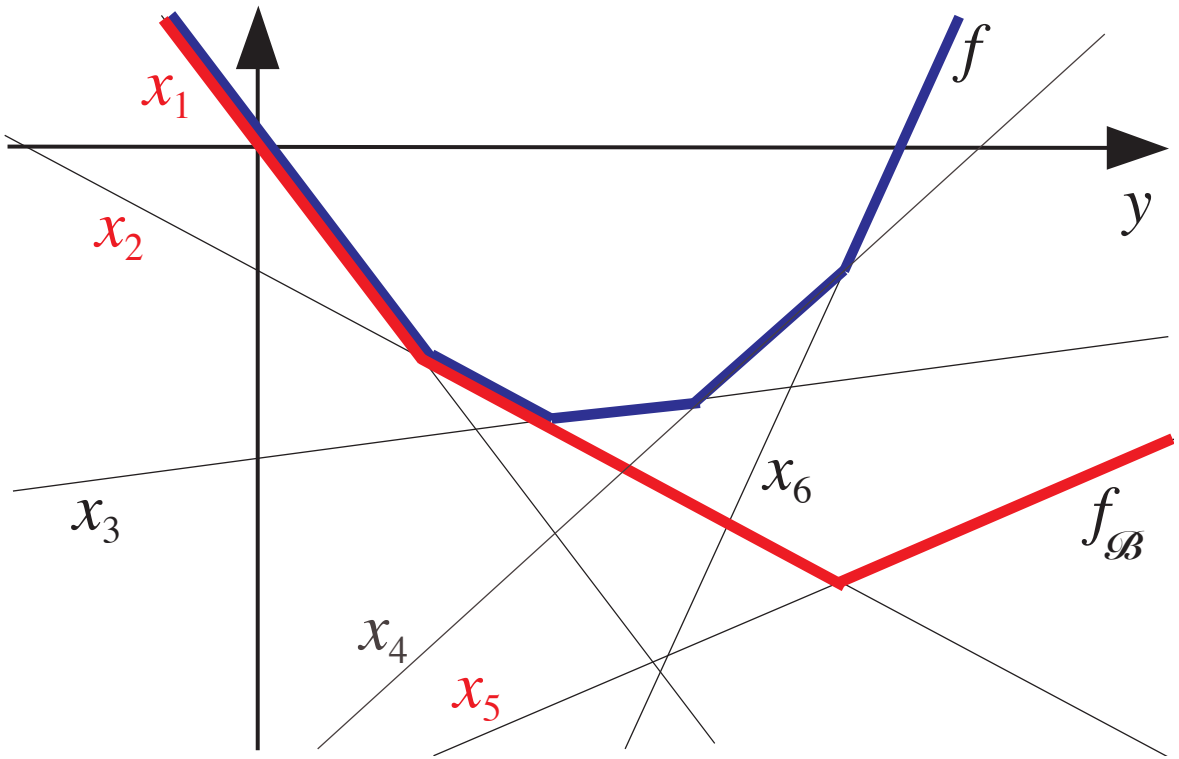
$$v(\Pi) \geq cx^* = y^*b + v^* \geq f(\,y^*\,) \geq v(\Delta) \geq v(\Pi) \implies$$

$x^*/(\,v^*\,,\,y^*\,)$ 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[3] (applied to (Δ))

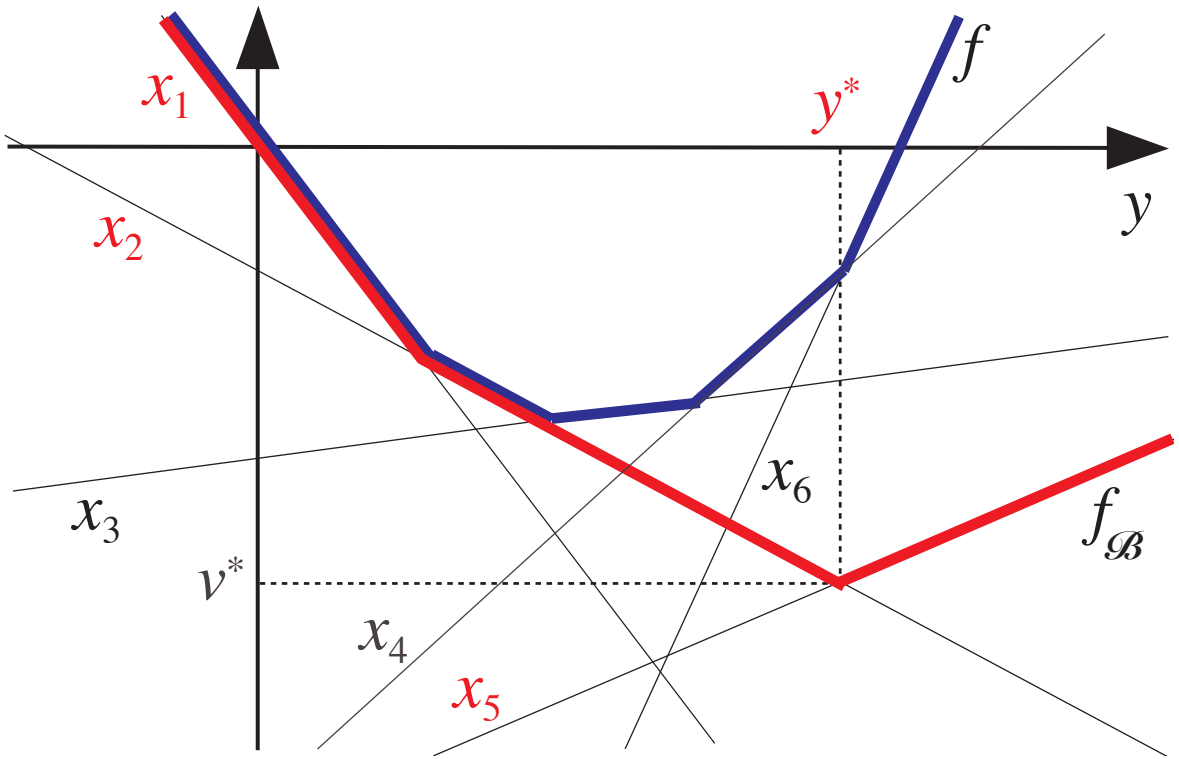
[3] Kelley "The Cutting-Plane Method for Solving Convex Programs" *J. of the SIAM*, 1960

Geometry of the Lagrangian dual



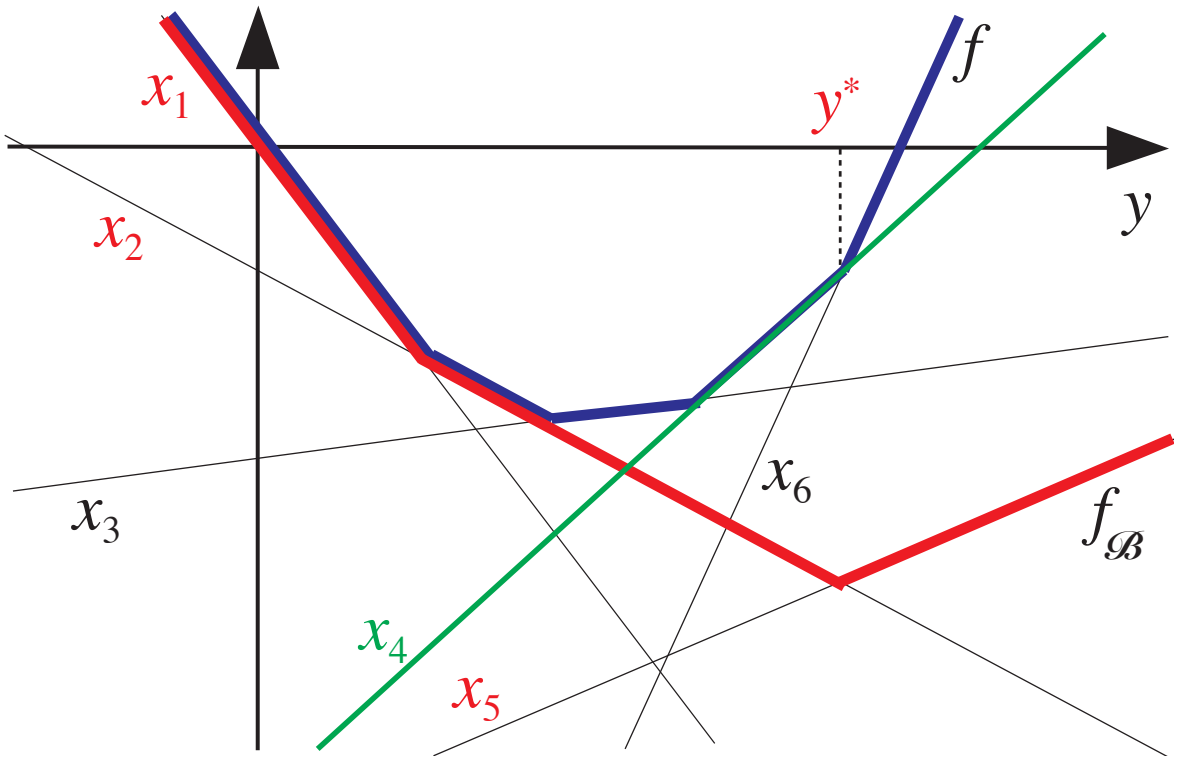
- $f_{\mathcal{B}} \leq f$ (CP model),

Geometry of the Lagrangian dual



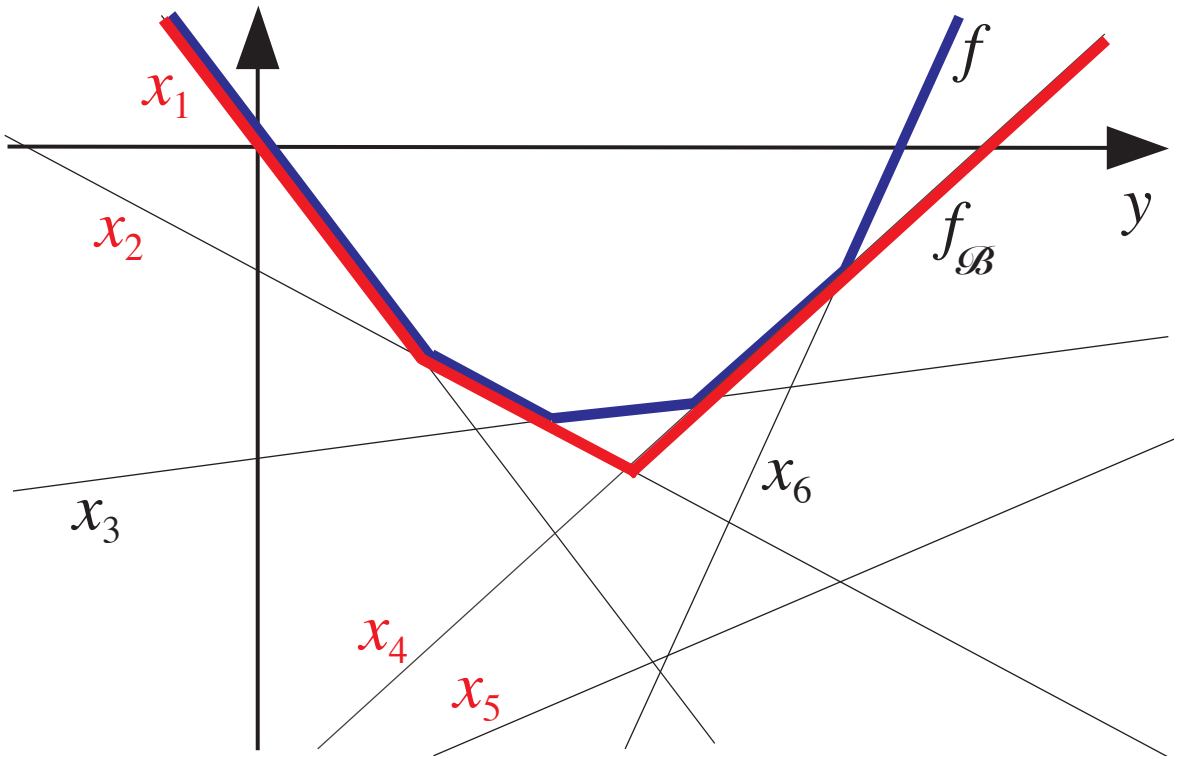
- $f_{\mathcal{B}} \leq f$ (CP model), $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$

Geometry of the Lagrangian dual



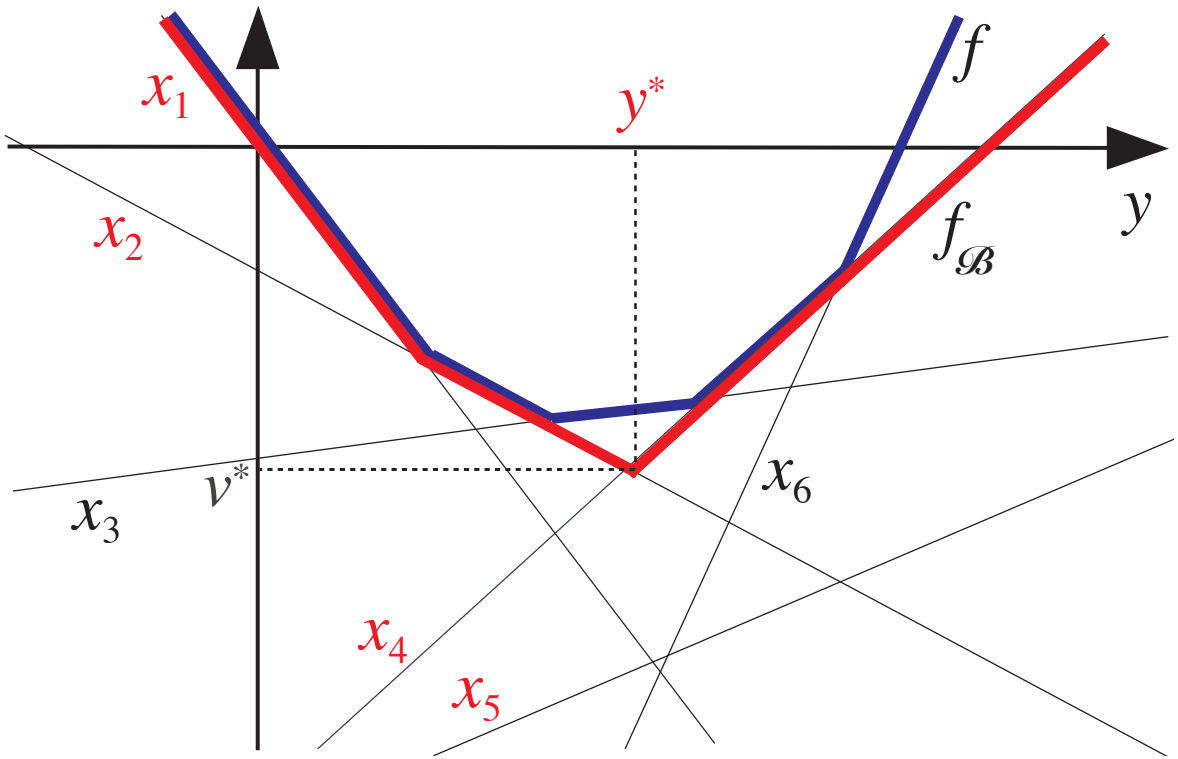
- $f_{\mathcal{B}} \leq f$ (CP model), $v^* = f_{\mathcal{B}}(\, y^* \,)$ lower bound on $v(\Pi_{\mathcal{B}})$
- Optimal solution $\bar{x}$ gives separator between $(\, v^* \,, \, y^* \,)$ and *epi* f $\equiv$ $(\, c\bar{x} \,, \, A\bar{x} \,) =$ new row in $(\Delta_{\mathcal{B}})$ (subgradient of f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

Geometry of the Lagrangian dual



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

Geometry of the Lagrangian dual



- $f_{\mathcal{B}} \leq f$ (CP model), $v^* = f_{\mathcal{B}}(\, y^* \,)$ lower bound on $v(\Pi_{\mathcal{B}})$
- Optimal solution $\bar{x}$ gives separator between $(\, v^* \,, \, y^* \,)$ and *epi* f $\equiv$ $(\, c\bar{x} \,, \, A\bar{x} \,)$ = new row in $(\Delta_{\mathcal{B}})$ (subgradient of f at y^*)
- Improve CP model, re-solve the master problem, rinse & repeat

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.,

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

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

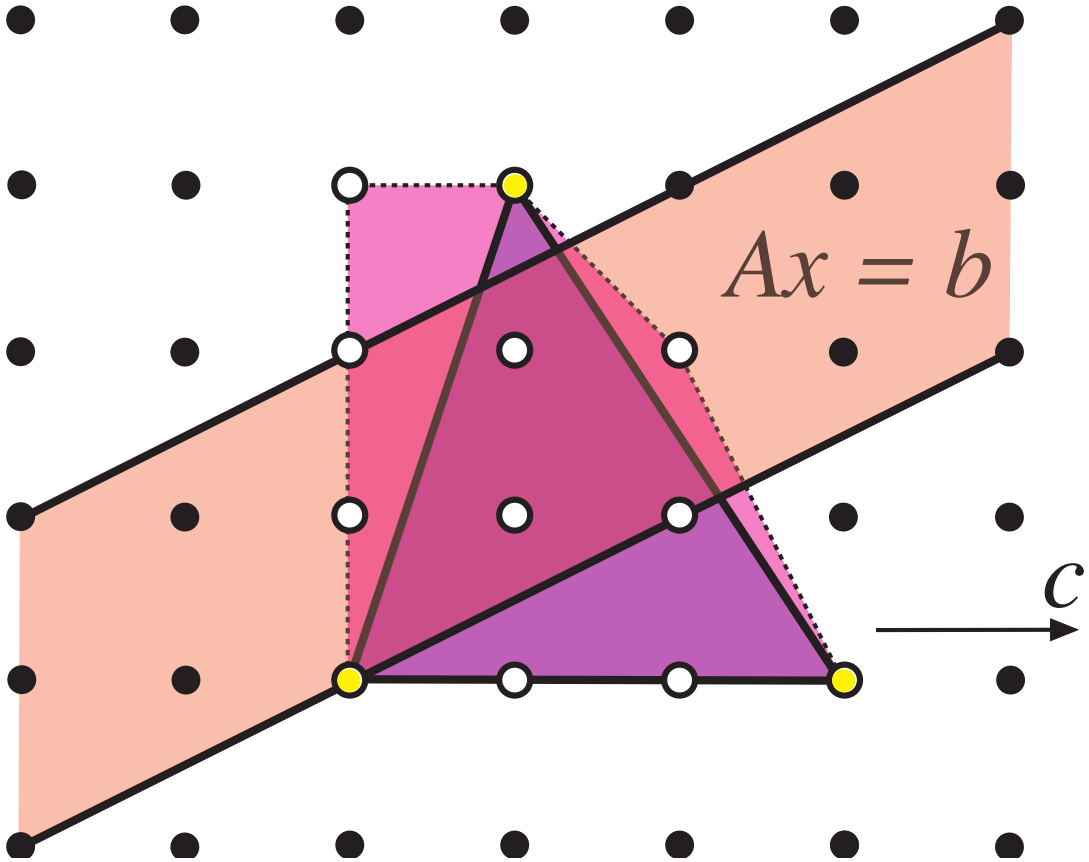
- $(c - y^*A)\bar{x} = \max\{ (c - y^*A)x \ : \ x \in X \} \geq (c - y^*A)x^*$

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

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

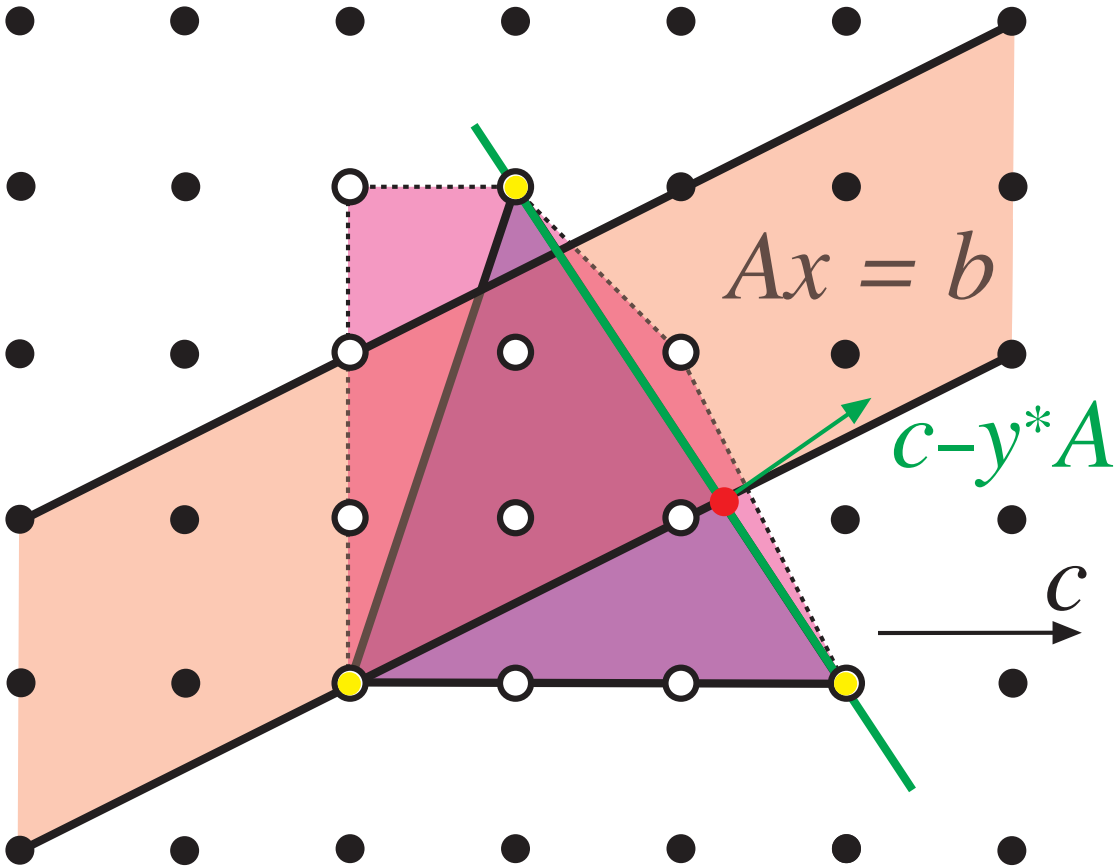
$\Longrightarrow \bar{x} \notin conv(\mathcal{B}) \Longrightarrow$ makes sense to add $\bar{x}$ to $\mathcal{B}$

Geometry of Dantzig-Wolfe/Column Generation



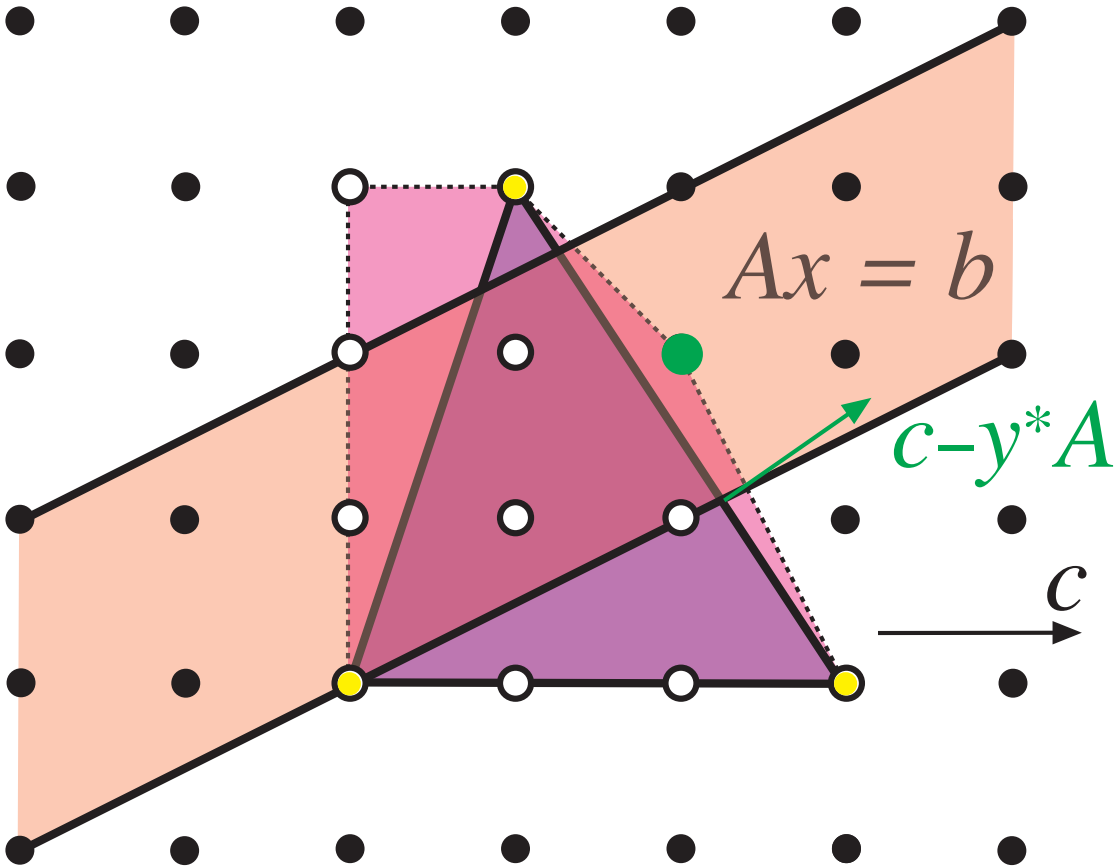
- $X_{\mathcal{B}} = conv(\mathcal{B})$ inner approximation of X

Geometry of Dantzig-Wolfe/Column Generation



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

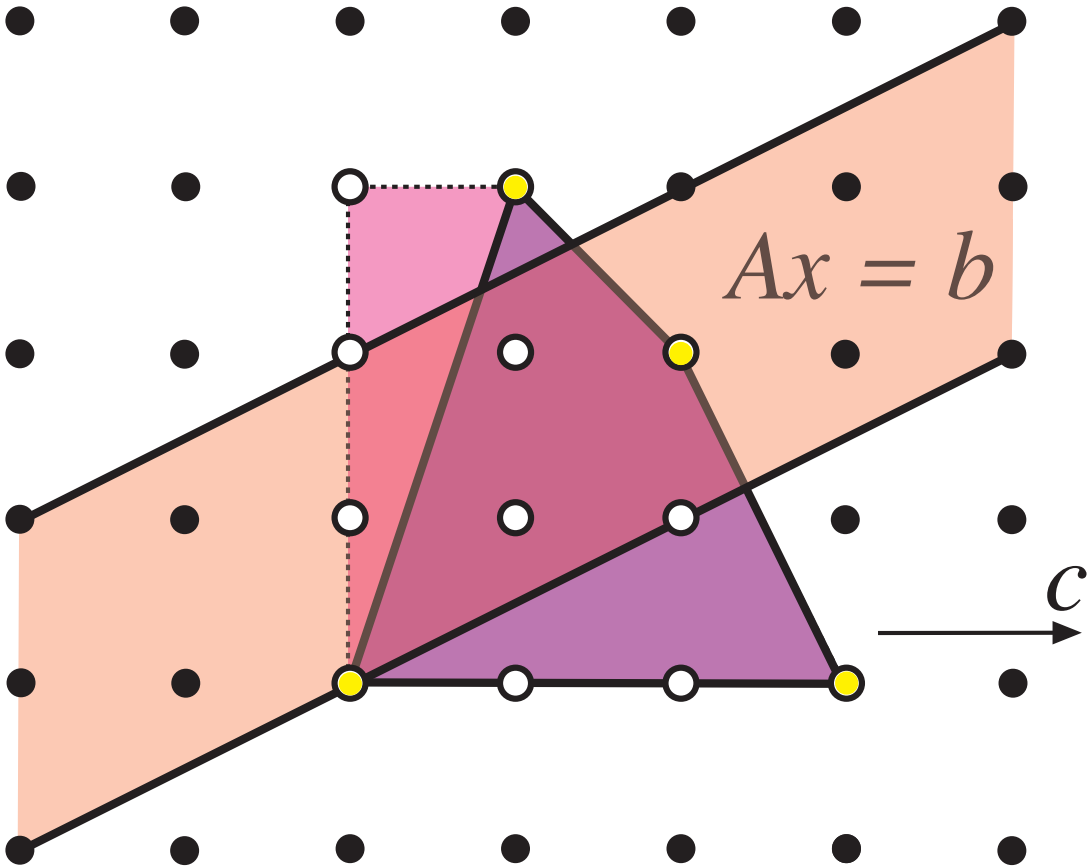
Geometry of Dantzig-Wolfe/Column Generation



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

  $\Longrightarrow$ optimizing $c - y^*A$ finds new $\bar{x} \in X$ (if any)

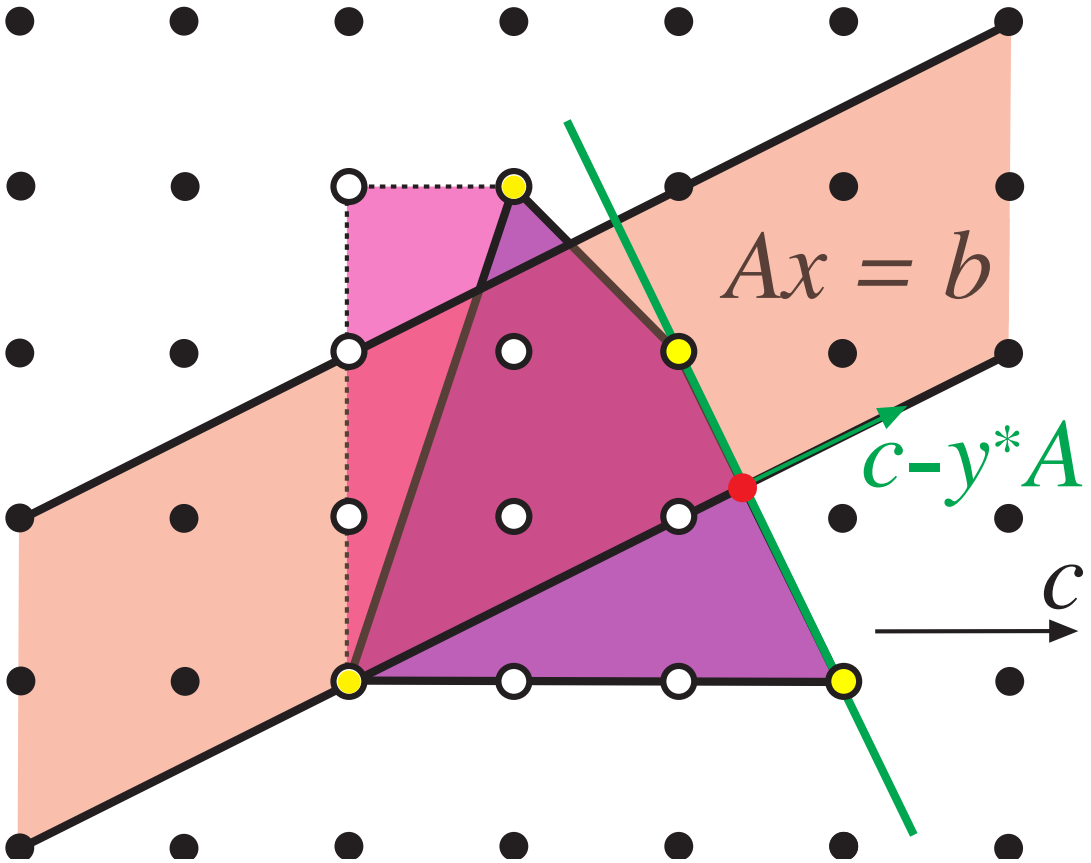
Geometry of Dantzig-Wolfe/Column Generation



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

  $\Longrightarrow$ optimizing $c - y^*A$ finds new $\bar{x} \in X$ (if any)
- Increase $X_{\mathcal{B}}$,

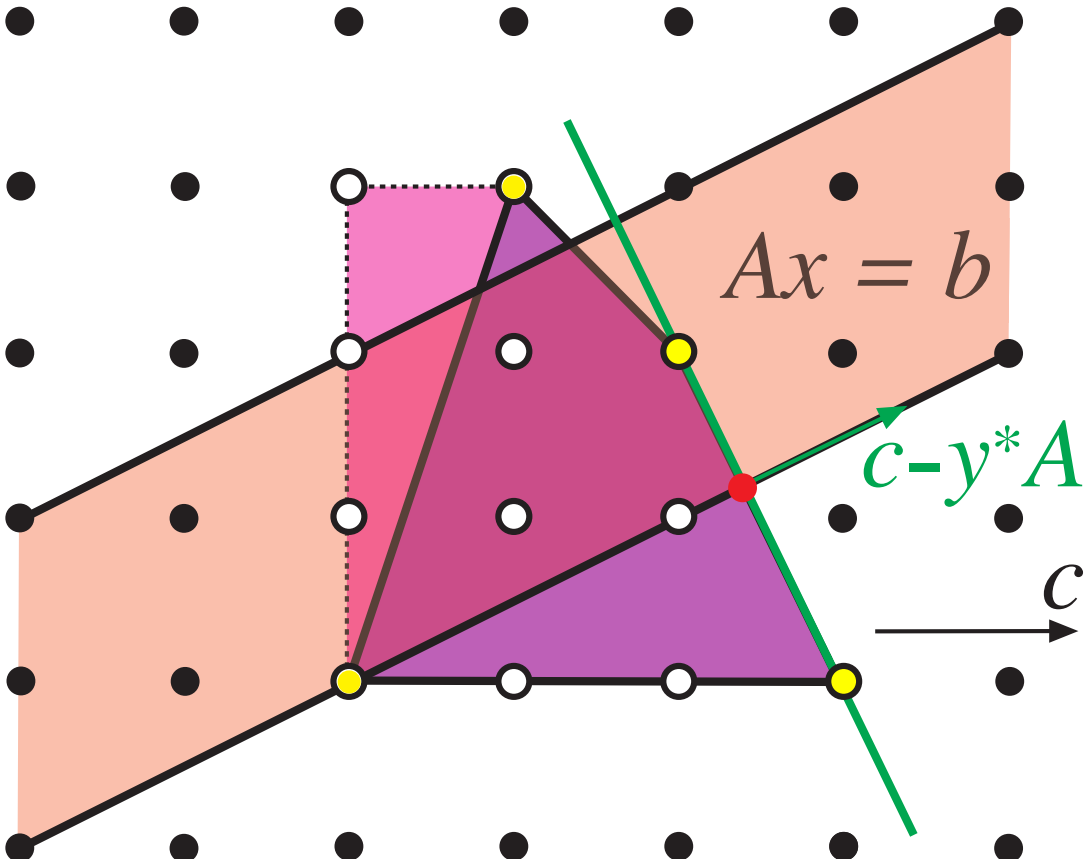
Geometry of Dantzig-Wolfe/Column Generation



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

  $\Longrightarrow$ optimizing $c - y^*A$ finds new $\bar{x} \in X$ (if any)
- Increase $X_{\mathcal{B}}$, re-solve master problem, rinse & repeat

Geometry of Dantzig-Wolfe/Column Generation



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

  $\Longrightarrow$ optimizing $c - y^*A$ finds new $\bar{x} \in X$ (if any)
- Increase $X_{\mathcal{B}}$, re-solve master problem, rinse & repeat
- Issue: $X_{\mathcal{B}} \cap Ax = b$ must be nonempty

The Unbounded Case

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

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

- 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[3]

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) $\Longrightarrow$

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

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

- 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

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

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

- Clever trick[4]: 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 before, $W = conv(\ ext\ W = W_0\ ) + cone(\ ext\ rec\ W = W_\infty\ ) \Longrightarrow$

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

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

[4] 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

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

$= \max \left\{ cx + v_{\mathcal{B}}(x) \ : \ x \in X \cap V_{\mathcal{B}} \right\}$, 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[3] to (B)
- 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

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

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

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

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):

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

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

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

  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[3] $\equiv$
  revised simplex method with mechanized pricing in the discrete case

- Natural $\not\Longrightarrow$ fast: convex nondifferentiable optimization $\Omega(1 / \varepsilon^2)$,
  Cutting-Plane method much worse than that (will see soon)

- Many variants/other algorithms possible, another story (course)

You can apply Lagrange to a Staircase-structured program

- Reformulate a staircase-structured program

$$\max\ cx + e'z' + e''z''$$

$$Dx + E'z' \leq d' \ , \ Dx + E''z'' \leq d''$$

$$x \in X$$

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
  (but beware the multi-stage case)

You can apply Benders' to a Block-diagonal program

- Reformulate a block-diagonal program

$$\max\ c'x' + c''x''$$

$$E'x' \leq d'\ ,\ E''x'' \leq d''$$

$$A'x' + A''x'' = b$$

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

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[5] in multicommodity parlance

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

Dual Decomposition: the Nonlinear and Integer Cases

Block-diagonal Convex Nonlinear Programs

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

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

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

- 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[6] (usually easy to get)

[6] 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

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

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)

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

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[7], 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)

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

Block-diagonal Integer Programs

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

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

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

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)

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

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

- $\theta_{\bar{x}} \in \mathbb{Z}$ gives a reformulation of (Π); could branch on $\theta_{\bar{x}}$, but usually better doing it on x, easier to integrate in the relaxation computation

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[8,9]
- 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[10] (for now)

[8] Daniilidis, Lemaréchal "On a Primal-Proximal Heuristic in Discrete Optimization" *Math. Prog.*, 2005

[9] Scuzziato, Finardi, F. "Solving Stochastic [...] Unit Commitment with a New Primal Recovery [...]" *IJEPES*, 2021

[10] https://scipopt.org, https://gcg.or.rwth-aachen.de

Digression: How to Choose your Lagrangian relaxation

- There may be many choices

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

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

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

- The best between (Δ') and (Δ'') depends on integrality of X', X'':
  - if both have it, both (Δ') and (Δ'') $\equiv$ continuous relaxation
  - if only one has it, the one that does not, but if both don't have it?
- Here comes Lagrangian decomposition[11] (looks familiar?)

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

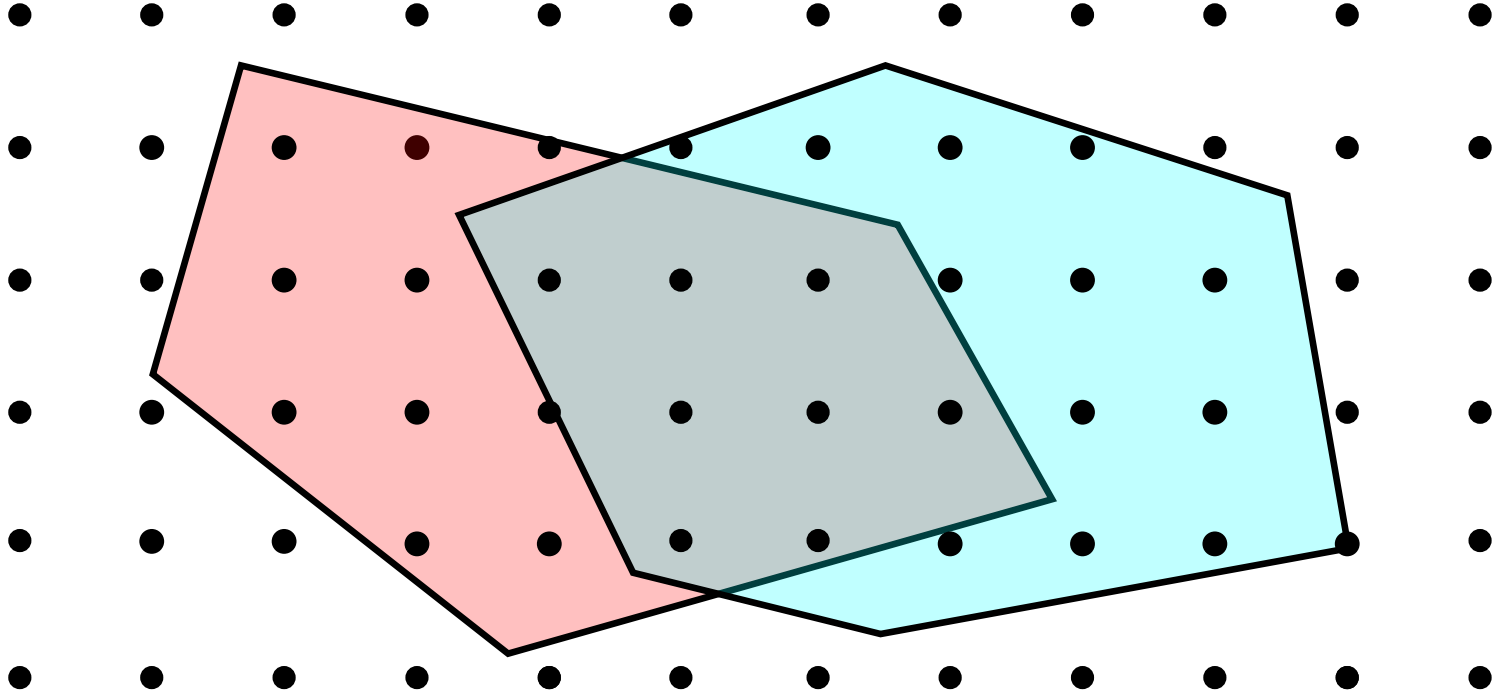
$$(\Pi_\lambda) \quad \max \{ (c/2 + \lambda)x' \ : \ x' \in X' \} + \max \{ (c/2 - \lambda)x'' \ : \ x'' \in X'' \}$$

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

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

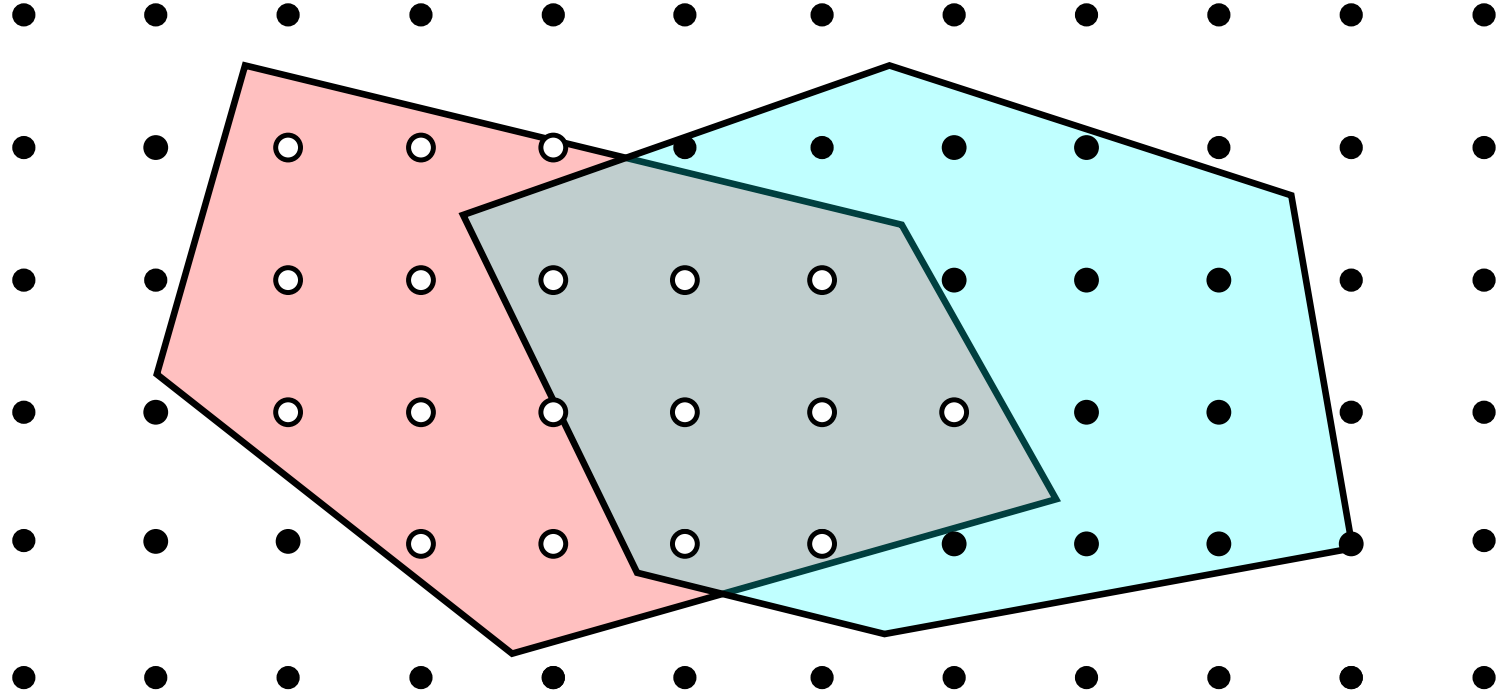
[11] Guignard, Kim "Lagrangean Decomposition: a Model Yielding Stronger Lagrangean Bounds" *Math. Prog.*, 1987

Geometry of Lagrangian Decomposition



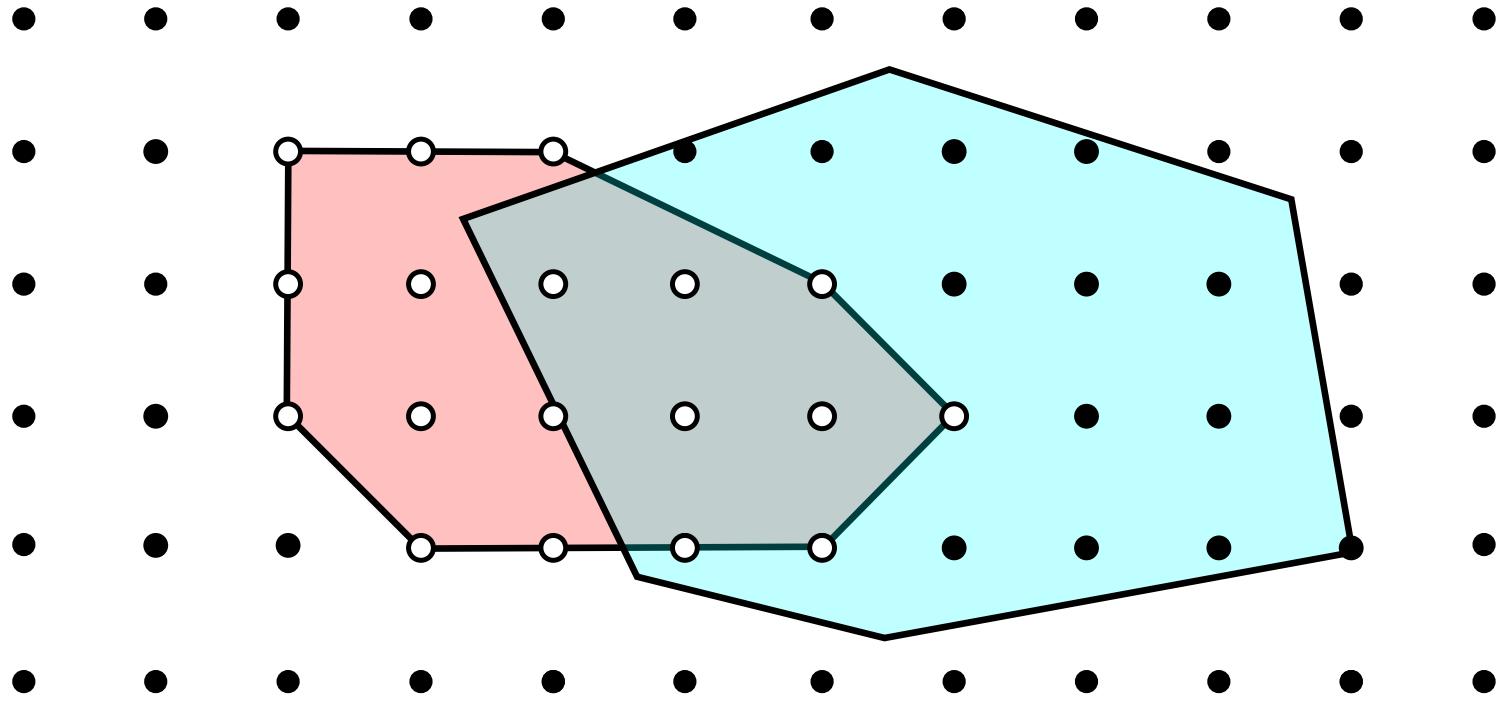
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation

Geometry of Lagrangian Decomposition



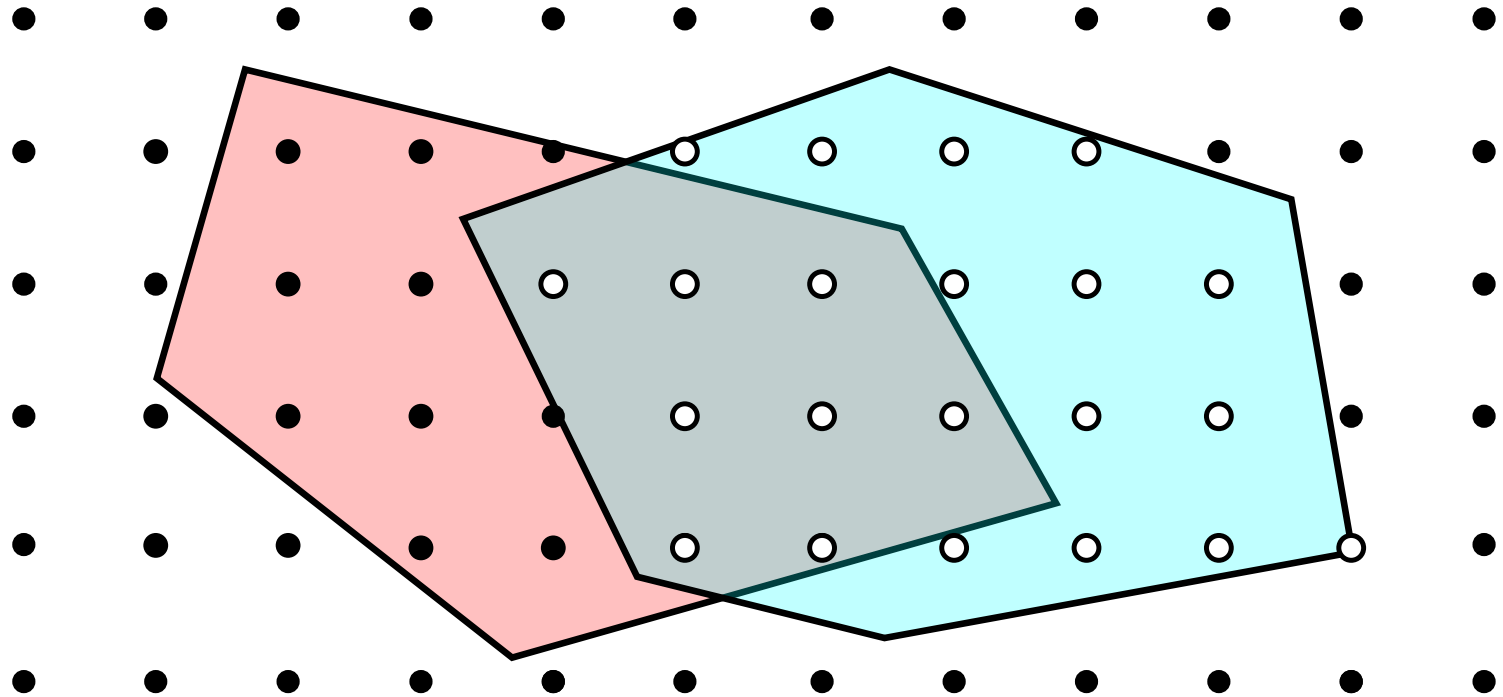
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints

Geometry of Lagrangian Decomposition



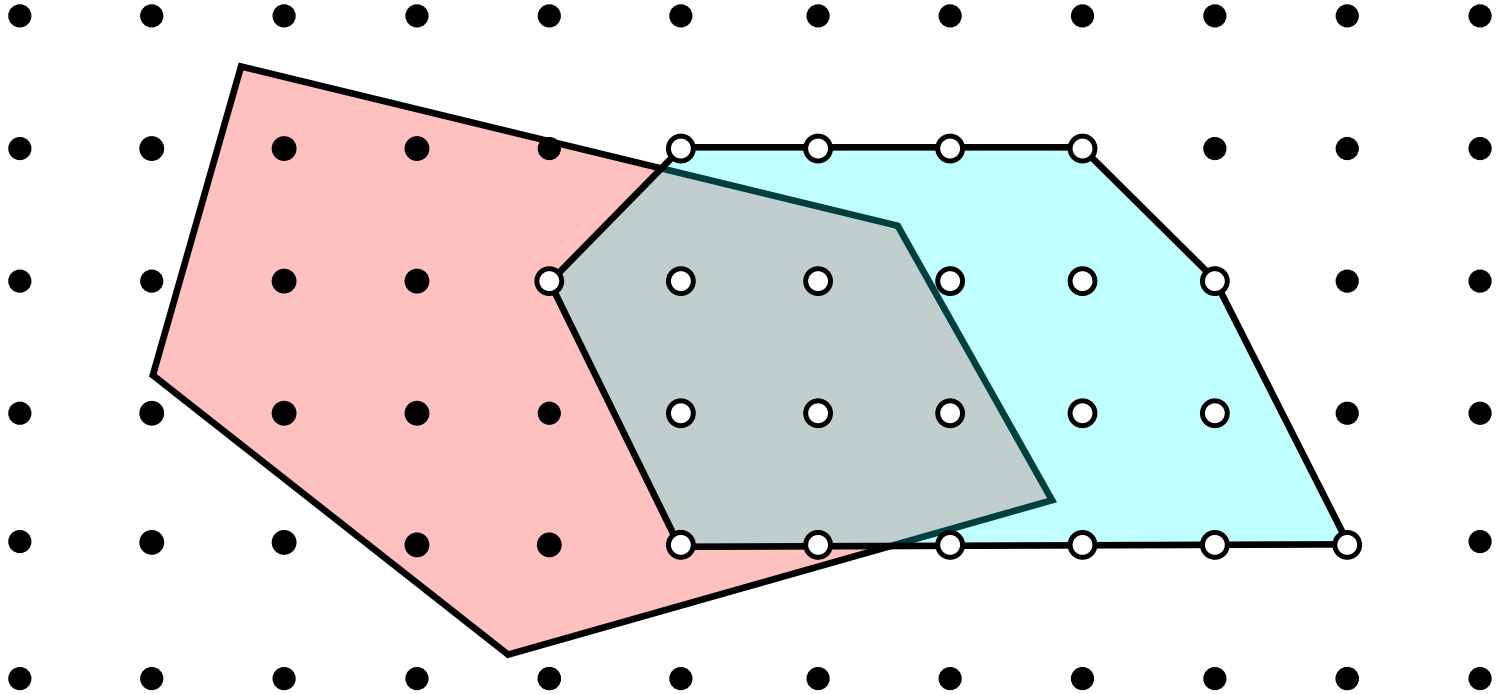
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part

Geometry of Lagrangian Decomposition



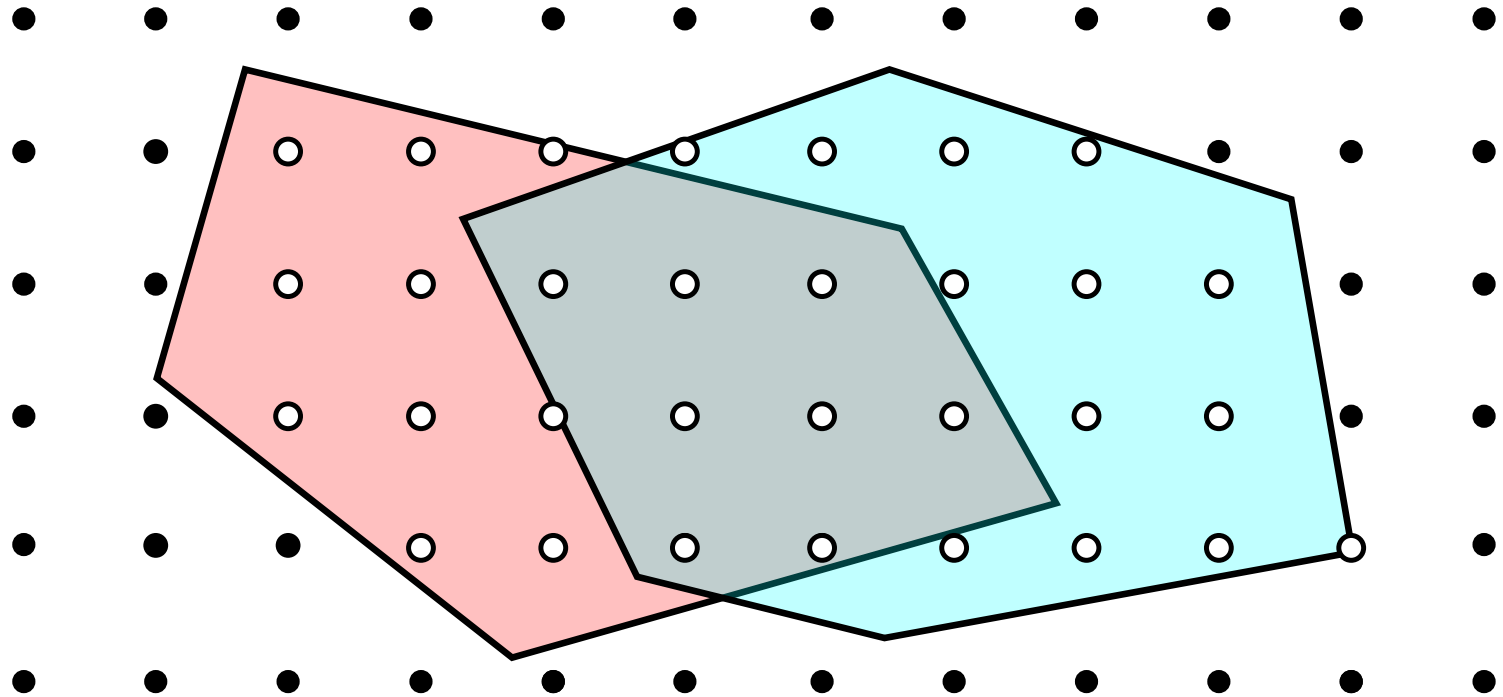
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints

Geometry of Lagrangian Decomposition



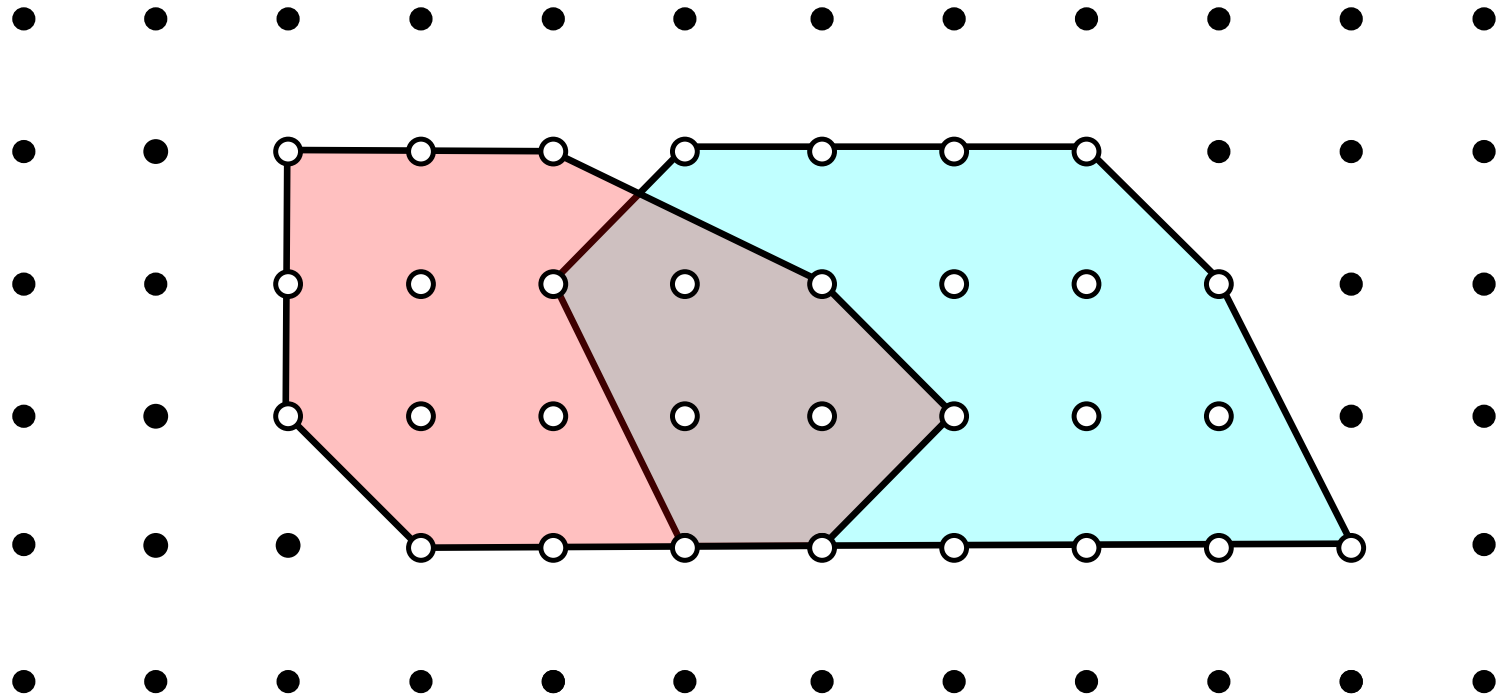
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part

Geometry of Lagrangian Decomposition



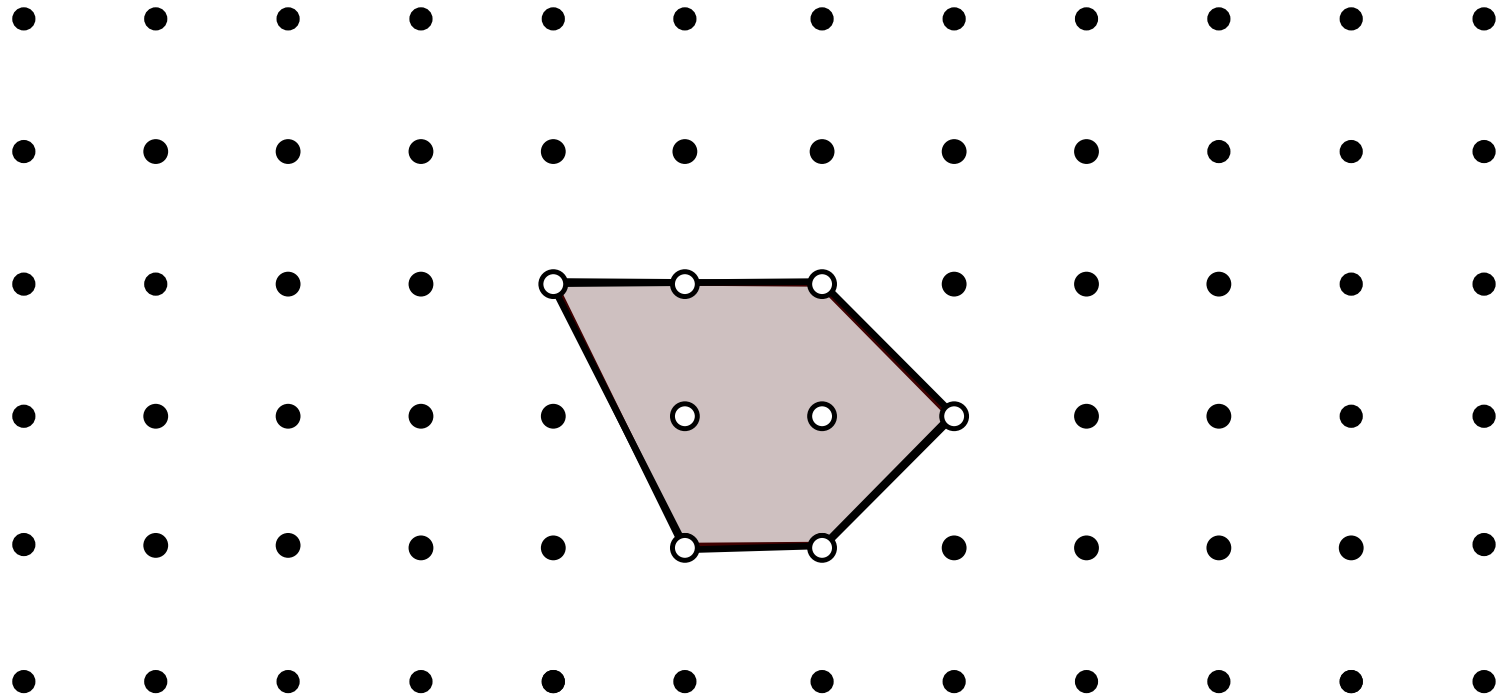
- Intersection between red and blue $\equiv$ grey $\equiv$ continuous relaxation
- Lagrangian relaxation of blue constraints shrinks the red ($\Longrightarrow$ grey) part
- Lagrangian relaxation of red constraints shrinks the blue ($\Longrightarrow$ grey) part
- Lagrangian decomposition (both red and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

A Computational Example: Capacitated Facility Location

- Set O of facilities to be installed, cost f_i and capacity u_i for $i \in O$
- Set D of customers to be served, demand d_j (unique product) for $j \in D$
- Unitary transport cost c_{ij} on arc $(i, j) \in A$ (facility $i \rightarrow$ customer j)

$$\min \sum_{(i,j)\in A} c_{ij} d_j x_{ij} + \sum_{i \in O} f_i z_i \tag{1}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \qquad j \in D \tag{2}$$

$$\sum_{j:(i,j)\in A} d_j x_{ij} \leq u_i z_i \qquad i \in O \tag{3}$$

$$x_{ij} \in [0,1] \,/\, \{0,1\} \qquad (i,j) \in A \tag{4}$$

$$z_i \in \{0,1\} \qquad i \in O \tag{5}$$

- Splittable / unsplittable: customers can/not be served by > 1 facility
- > 1 products $\rightarrow$ multicommodity network design with very simple paths

Lagrangian Relaxations of Capacitated Facility Location

- Relax (2): $|O|$ (mixed-integer) knapsacks

$$\sum_{j\in D} y_j + \min \sum_{i\in O} \Big[\sum_{j:(i,j)\in A} (c_{ij} d_j - y_j) x_{ij} + f_i z_i \Big] \tag{6}$$

$$\sum_{j:(i,j)\in A} d_j x_{ij} \le u_i z_i \qquad i \in O \tag{3}$$

$$x_{ij} \in [0,1] \,/\, \{0,1\} \qquad (i,j) \in A \tag{4}$$

$$z_i \in \{0,1\} \qquad i \in O \tag{5}$$

- Relax (3): $|O|$ 1-variable problems $+|D|$ simple choice problems

$$\min \sum_{j\in D} \sum_{i:(i,j)\in A} d_j (c_{ij} + w_i) x_{ij} + \sum_{i\in O} (f_i - w_i u_i) z_i \tag{7}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \qquad j \in D \tag{2}$$

$$x_{ij} \in [0,1] \,/\, \{0,1\} \qquad (i,j) \in A \tag{4}$$

$$z_i \in \{0,1\} \qquad i \in O \tag{5}$$

Exercise: which relaxation gives the best bound in the splittable / unsplittable case?

Dantzig-Wolfe Reformulation $\equiv$ Column Generation

- Column-generation view of the problem: patterns for facility i

  $\mathcal{P}^i = \{\, p \in [0,1]^{|D|} \;:\; \sum_{j \in D} d_j p_j \leq u_i \;,\; (i\,,\,j) \notin A \Longrightarrow p_j = 0 \,\}$

  except $p = 0$, $+$ integrality if needed

- $p \in \mathcal{P}^i \Longrightarrow c_p = f_i + \sum_{j\,:\,(i\,,j) \in A} c_{ij} d_j p_j$, $\mathcal{P} = \bigcup_{i \in O} \mathcal{P}^i$

- (disaggregated) Dantzig-Wolfe reformulation $\equiv$

$$\min \sum_{i \in O} \sum_{p \in \mathcal{P}^i} c_p \theta_p \tag{8}$$

$$\sum_{p \in \mathcal{P}} p_j \theta_p = 1 \qquad j \in D \tag{9}$$

$$\sum_{p \in \mathcal{P}^i} \theta_p \leq 1 \qquad i \in O \tag{10}$$

$$\theta_p \geq 0 \qquad p \in \mathcal{P} \tag{11}$$

- D-W/CP: start with (small) $\mathcal{B}^i \subset \mathcal{P}^i$, solve (8)–(11) restricted to $\mathcal{B}$, take y_i duals of (9), solve Lagrangian relaxations, rinse & repeat

- Eventually yields good bounds . . .

Dantzig-Wolfe Reformulation $\equiv$ Column Generation

- Column-generation view of the problem: patterns for facility i

  $\mathcal{P}^i = \{\, p \in [0,1]^{|D|} \;:\; \sum_{j \in D} d_j p_j \leq u_i \;,\; (i\,,\,j) \notin A \Longrightarrow p_j = 0 \,\}$

  except $p = 0$, $+$ integrality if needed

- $p \in \mathcal{P}^i \Longrightarrow c_p = f_i + \sum_{j\,:\,(i\,,j) \in A} c_{ij} d_j p_j$, $\mathcal{P} = \bigcup_{i \in O} \mathcal{P}^i$

- (disaggregated) Dantzig-Wolfe reformulation $\equiv$

$$\min \sum_{i \in O} \sum_{p \in \mathcal{P}^i} c_p \theta_p \tag{8}$$

$$\sum_{p \in \mathcal{P}} p_j \theta_p = 1 \qquad j \in D \tag{9}$$

$$\sum_{p \in \mathcal{P}^i} \theta_p \leq 1 \qquad i \in O \tag{10}$$

$$\theta_p \geq 0 \qquad p \in \mathcal{P} \tag{11}$$

- D-W/CP: start with (small) $\mathcal{B}^i \subset \mathcal{P}^i$, solve (8)–(11) restricted to $\mathcal{B}$, take y_i duals of (9), solve Lagrangian relaxations, rinse & repeat

- Eventually yields good bounds . . . if the master problem is nonempty

Algorithmic Issues

- Issue: the master problem can be (primal) empty ($\equiv$ dual unbounded)
- Phase 0 approach: seek for feasible solution first

$$\min \sum_{j \in D} v_j \tag{12}$$

$$\sum_{p \in \mathcal{B}} p_j \theta_p + v_j = 1 \qquad j \in D \tag{13}$$

$$\sum_{p \in \mathcal{B}^i} \theta_p \leq 1 \qquad i \in O \tag{10}$$

$$\theta_p \geq 0 \qquad p \in \mathcal{B} \tag{11}$$

$$v_j \geq 0 \qquad j \in D \tag{14}$$

- Minimise cost of slack variables v_j, disregard true costs
- Ends with some $v_j^* > 0 \equiv$ DW reformulation $\Longrightarrow$ original problem empty
- Otherwise master problem feasible with $\mathcal{B}$, start "true" optimization

Algorithmic Issues

- Issue: the master problem can be (primal) empty ($\equiv$ dual unbounded)
- Phase 0 approach: seek for feasible solution first

$$\min \sum_{j \in D} v_j \tag{12}$$

$$\sum_{p \in \mathcal{B}} p_j \theta_p + v_j = 1 \qquad j \in D \tag{13}$$

$$\sum_{p \in \mathcal{B}^i} \theta_p \leq 1 \qquad i \in O \tag{10}$$

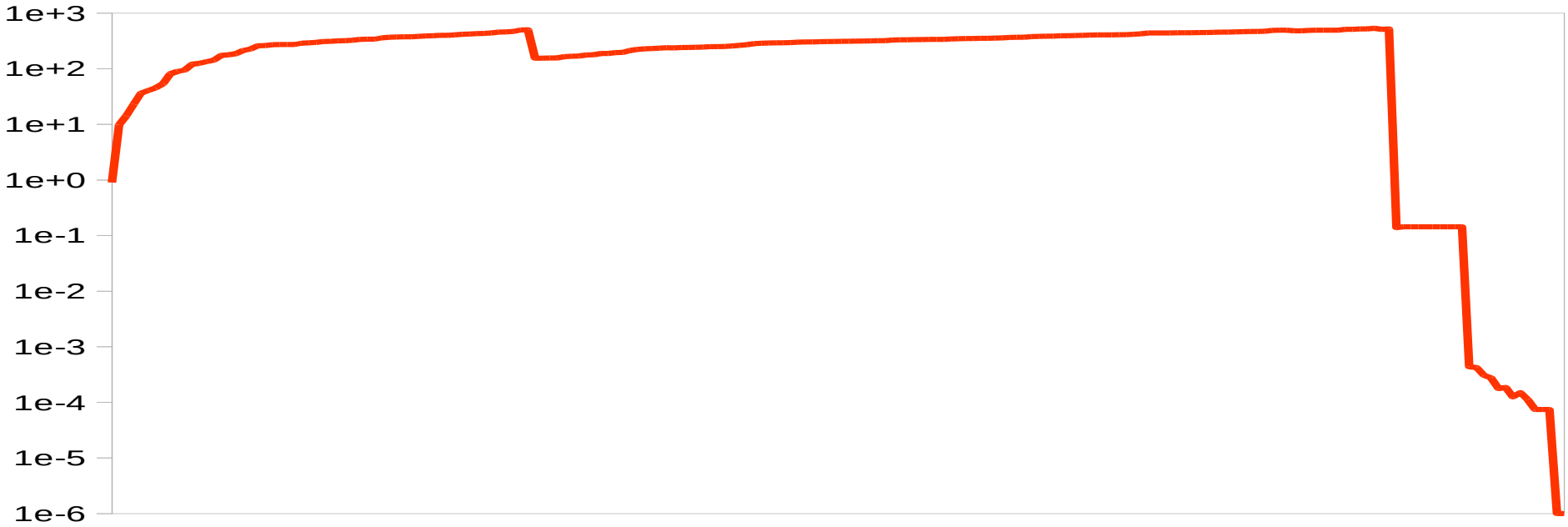
$$\theta_p \geq 0 \qquad p \in \mathcal{B} \tag{11}$$

$$v_j \geq 0 \qquad j \in D \tag{14}$$

- Minimise cost of slack variables v_j, disregard true costs
- Ends with some $v_j^* > 0 \equiv$ DW reformulation $\Longrightarrow$ original problem empty
- Otherwise master problem feasible with $\mathcal{B}$, start "true" optimization
- Real issue: can take forever because D-W/CP inefficient
- And you have to do branching (Branch & Price) on top of that

Don't try this at home, by-the book

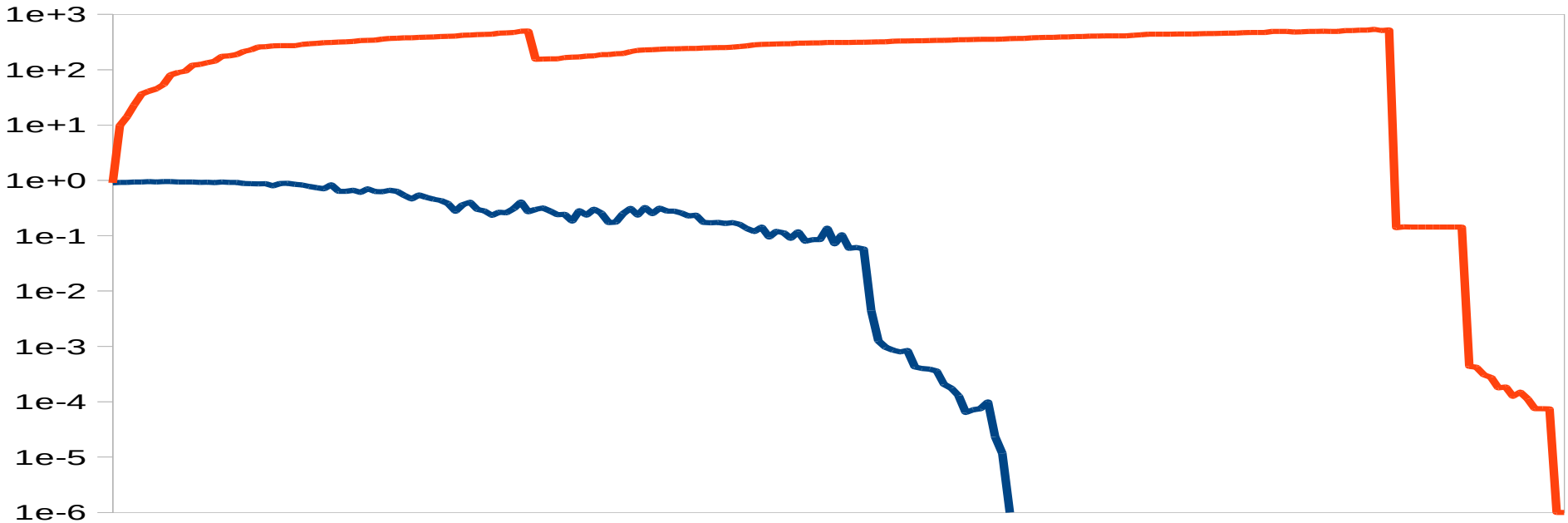
- How a by-the-book implementations behave:



- y^* immediately shoots much farther from optimum than initial point $\equiv$ having good initial point not much useful
- No apparent improvement for a long time as information slowly accrues
- A mysterious threshold is hit and "real" convergence begins

Don't try this at home, by-the book

- How a by-the-book implementations behave:

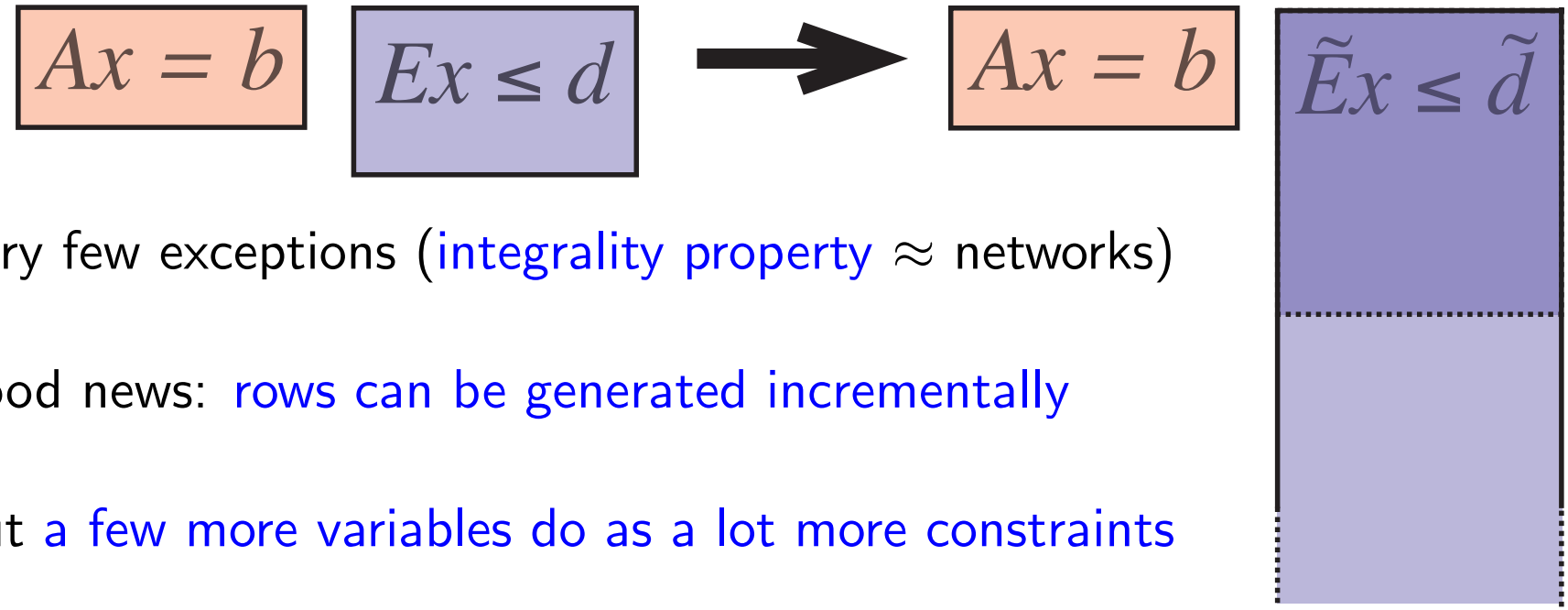


- y^* immediately shoots much farther from optimum than initial point
  $\equiv$ having good initial point not much useful
- No apparent improvement for a long time as information slowly accrues
- A mysterious threshold is hit and "real" convergence begins
- Can be improved (stablised), but that's another story (course)

Alternative Good Formulations for $conv(X)$

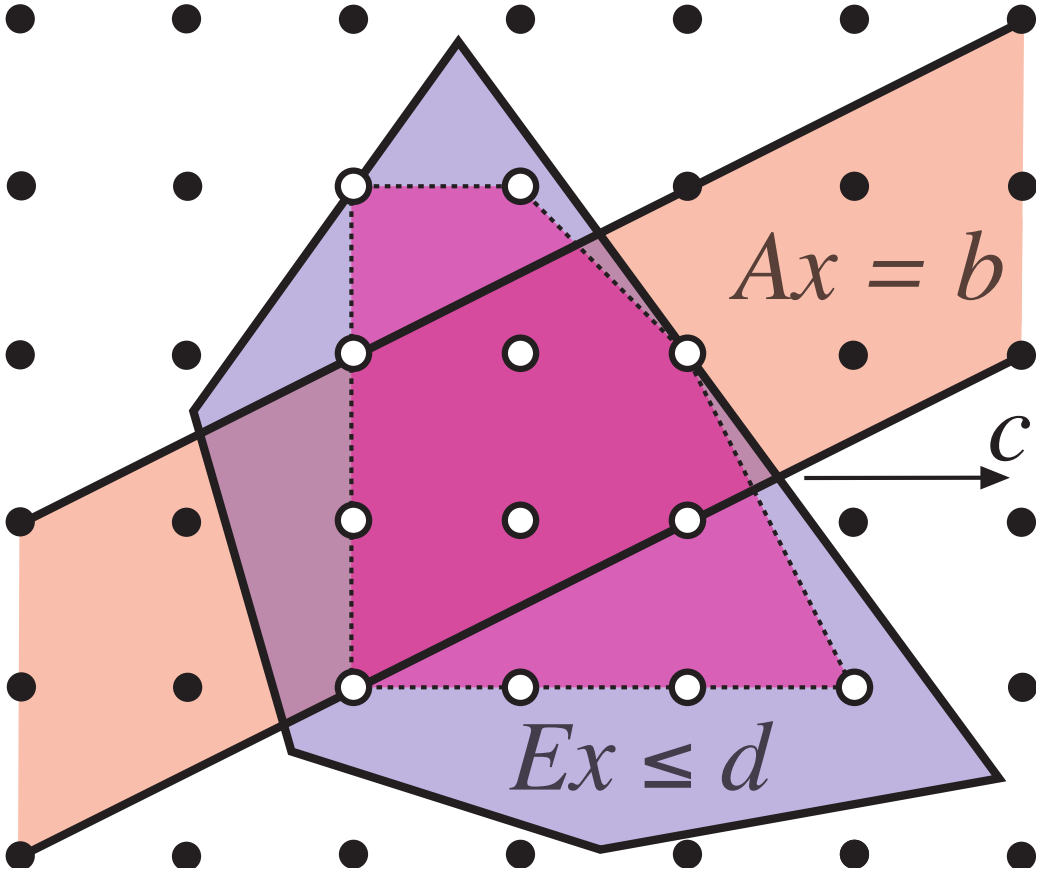
Alternative Good Formulations for $conv(X)$

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



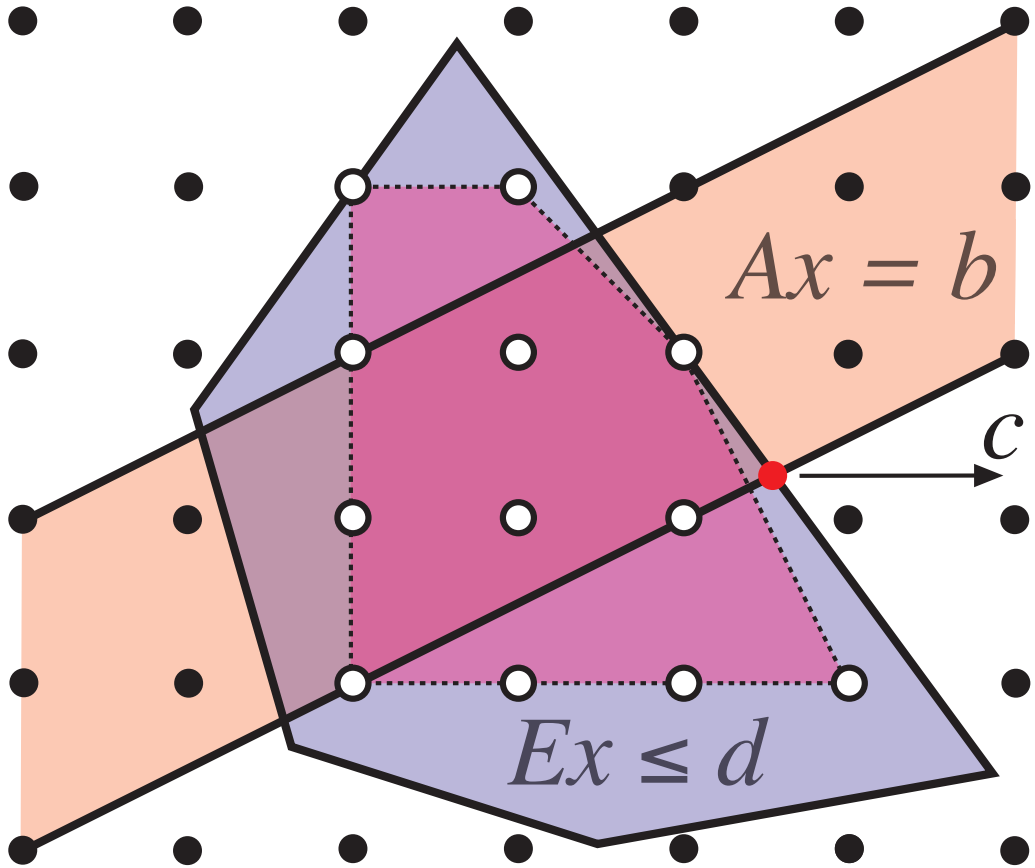
- Very few exceptions (integrality property $\approx$ networks)
- Good news: rows can be generated incrementally
- But a few more variables do as a lot more constraints

Row generation/polyhedral approaches



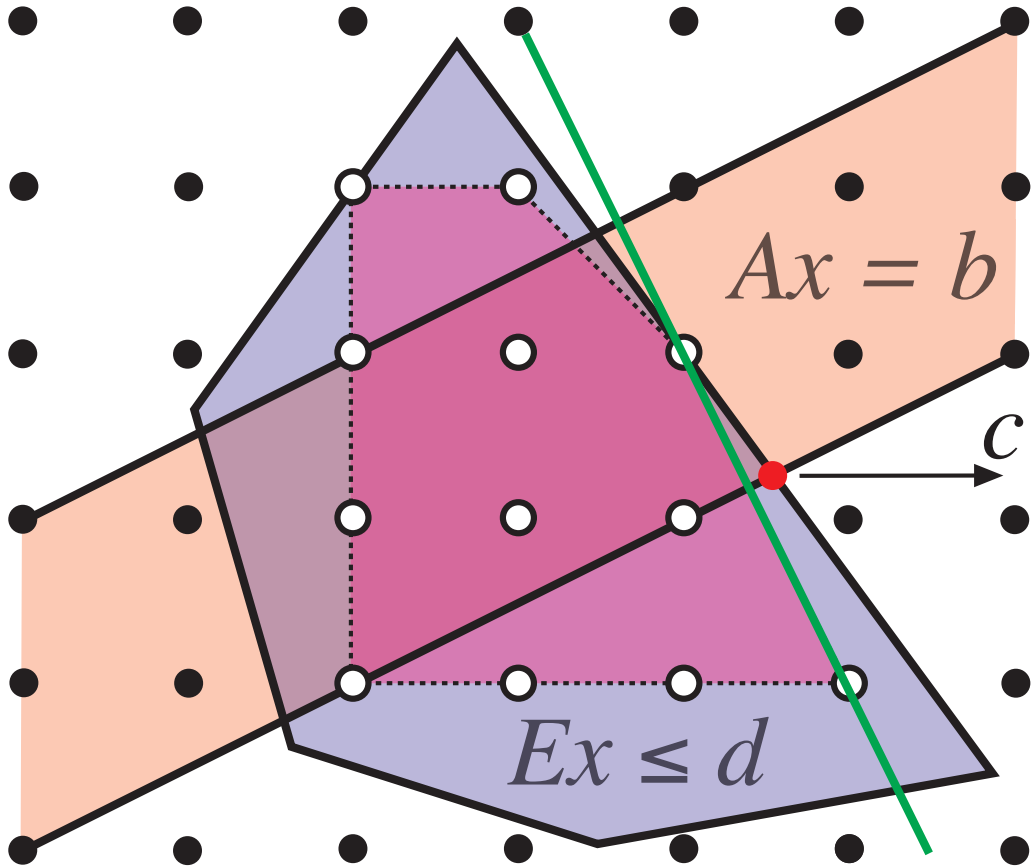
- $Ax \leq b \cap Ex \leq d$ outer approximation of feasible region

Row generation/polyhedral approaches



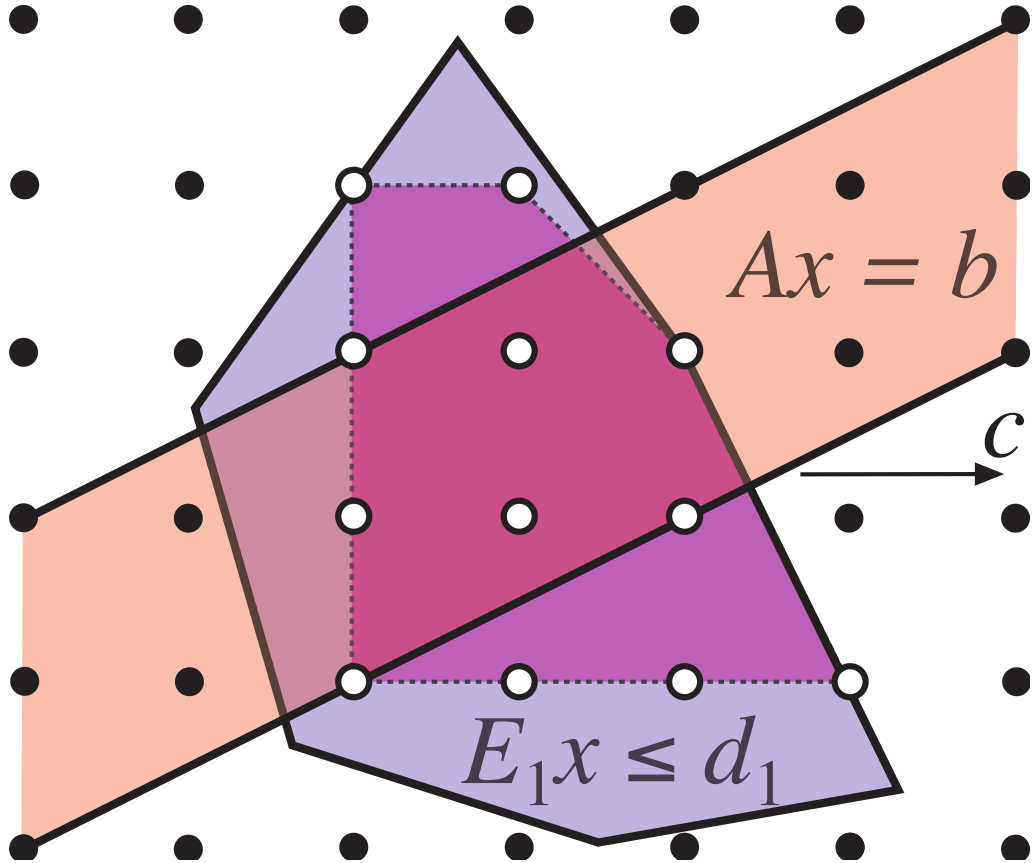
- $Ax \leq b \cap Ex \leq d$ outer approximation of feasible region
- Optimal solution of continuous relaxation gives bound,

Row generation/polyhedral approaches



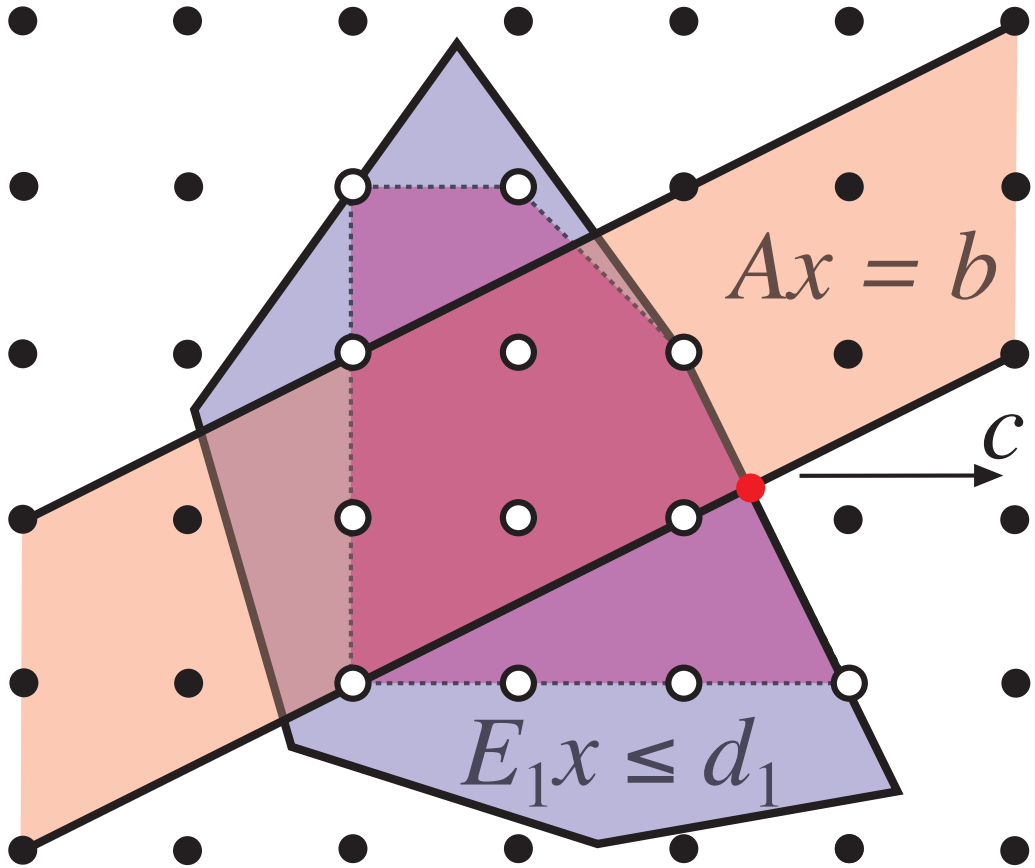
- $Ax \leq b \cap Ex \leq d$ outer approximation of feasible region
- Optimal solution of continuous relaxation gives bound, valid inequality $\equiv$ separator from $conv(X)$

Row generation/polyhedral approaches



- $Ax \leq b \cap Ex \leq d$ outer approximation of feasible region
- Optimal solution of continuous relaxation gives bound, valid inequality $\equiv$ separator from $conv(X)$
- Smaller feasible region,

Row generation/polyhedral approaches



- $Ax \leq b \cap Ex \leq d$ outer approximation of feasible region
- Optimal solution of continuous relaxation gives bound, valid inequality $\equiv$ separator from $conv(X)$
- Smaller feasible region, re-solve continuous relaxation, rinse & repeat

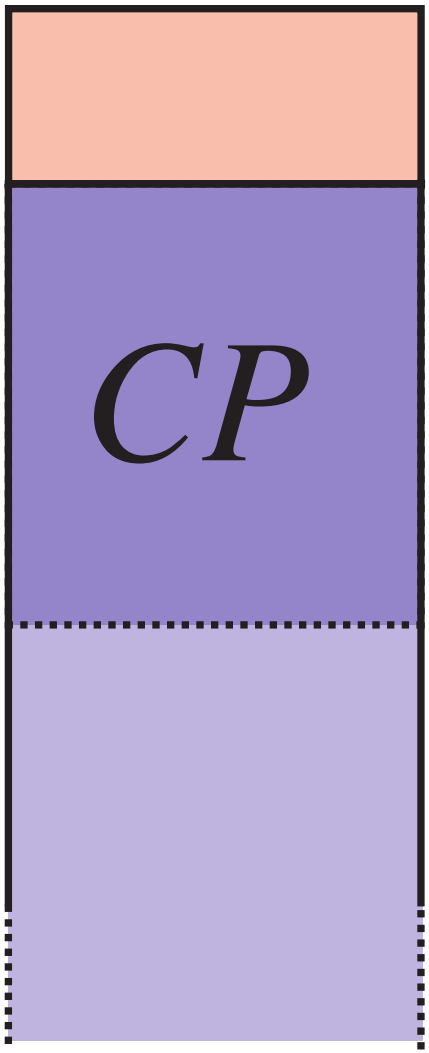
Example: Capacitated Facility Location

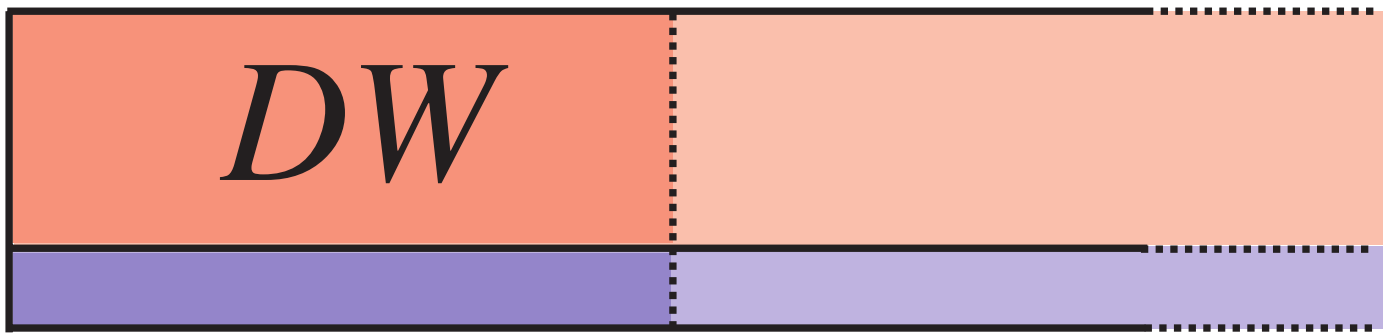
- Strong forcing constraints for Capacitated Facility Location

$$\begin{aligned} &\min\ (1) \\ &\quad (2)\,,\ (3)\,,\ (4)\,,\ (5) \\ &\quad x_{ij} \leq d_j z_i \qquad\qquad (i\,,\ j) \in A \end{aligned} \tag{15}$$

- Obviously valid, "only" $\#A$ many $\Longrightarrow$ trivially separable
- $\#A$ more constraints can make continuous relaxation unbearably slower
  $\Longrightarrow$ much better to separate them on-the-fly
- Just lazy constraints for solvers that support the notion
- Theoretical result: (15) $\Longrightarrow$ same bound as DW (in the splittable case)
- Many different ways to skin a cat (don't do this at home!)

A picture is worth 100 words





A Computational Example: CP vs. DW for CFL

Let's see some code running

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[12]
- May tail off, branching useful far before having solved $(\bar{\Pi}_X)$

[12] 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)

Primal Decomposition: the Nonlinear and Integer Cases

Staircase-structured z-convex Nonlinear Programs

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

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

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

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

$= \text{value function of a convex program} \Longrightarrow \text{convex}$

- $(B) \equiv (\Pi)$ without assumptions on $f(\cdot, z)$, $G(\cdot, z)$ and X, i.e., if (Π) is hard, then (B) is just as hard as (Π)
- (B) may still be more efficient (e.g., x "very few" but z "very many")
- Standard example: $X = \{ x \in \mathbb{Z}^n : Ex \leq d \}$ 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 $(\Pi) \Longrightarrow$ no branching needed unless for solving $(B_{\mathcal{B}})$

Staircase-structured z-convex Nonlinear Programs (cont.d)

- Still need duality: which one? Lagrangian[13], 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 $v(x^*) = -\infty \implies \{ z \in Z \ : \ G(x^*,z) \geq 0 \} = \emptyset \implies$
    $\exists\, \bar{\nu} \geq 0\,,\ \bar{z} \in Z$ s.t. $\max\{ \bar{\nu} G(x^*,z) \ : \ z \in Z \} = \bar{\nu} G(x^*,\bar{z}) < 0$

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

Staircase-structured z-convex Nonlinear Programs (cont.d)

- Still need duality: which one? Lagrangian[13], 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 $v(x^*) = -\infty \implies \{ z \in Z \ : \ G(x^*,z) \geq 0 \} = \emptyset \implies$
    $\exists\, \bar{\nu} \geq 0\,,\ \bar{z} \in Z$ s.t. $\max\{ \bar{\nu} G(x^*,z) \ : \ z \in Z \} = \bar{\nu} G(x^*,\bar{z}) < 0$

- General form of the master problem

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

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

Staircase-structured z-convex Nonlinear Programs (cont.d)

- Still need duality: which one? Lagrangian[13], 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 $v(x^*) = -\infty \implies \{ z \in Z \ : \ G(x^*,z) \geq 0 \} = \emptyset \implies$
    $\exists\, \bar{\nu} \geq 0\,,\ \bar{z} \in Z$ s.t. $\max\{ \bar{\nu} G(x^*,z) \ : \ z \in Z \} = \bar{\nu} G(x^*,\bar{z}) < 0$
- General form of the master problem

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

- Beware those nasty "max": must be that the "max" is independent of x!
- Possible in a few cases, complicated in general

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

Working Staircase-structured z-convex Nonlinear Programs

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

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

convex $\iff$ $f(\cdot)$ convex and $G(\cdot)$ concave ($\bar{\lambda} \geq 0$, $\bar{\nu} \geq 0$),
otherwise nonlinear nonconvex cuts, (B) "hard" (but (Π) was)

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

Staircase-structured non convex Nonlinear Programs

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

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

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

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

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

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: no Benders' for

$$(\Pi) \quad \max \{ cx + ez \ : \ Ax + Bz \leq b \ , \ x \in X \ , \ z \in \mathbb{Z}^m \}$$

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

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

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

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

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: no Benders' for
  $$(\Pi) \quad \max \{ cx + ez \ : \ Ax + Bz \leq b \ , \ x \in X \ , \ z \in \mathbb{Z}^m \}$$
- Some workarounds possible:
  - Use exact duality for nonconvex / integer problems[14] (though!)
  - Approximate the convex hull by some hierarchy[15] (RLT, ...)
  - Give up duality and use combinatorial Benders' (feasibility) cuts[16]
- In general much harder / less efficient
- Alternative route: use Benders' to solve continuous relaxation:
  Benders' subproblem as yet another (strong[17]) cut generator
- Often more efficient and supported by some off-the-shelf solver

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

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

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

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

Outline

1. Dual decomposition (Dantzig-Wolfe/Lagrangian/Column Generation)
2. Primal decomposition (Benders'/Resource)
3. All Are One, One Is All
4. Dual Decomposition: the Nonlinear and Integer Cases
5. Alternative Good Formulations for $conv(X)$
6. Primal Decomposition: the Nonlinear and Integer Cases
7. Decomposition-aware modelling systems
8. Conclusions (for now)

Modelling languages, and what they are for

- Most interactions with optimization solvers via Algebraic Modelling Languages (AML): commercial `AMPL` or `GAMS`[18], `AIMMS`[19] and `OPL`[20], or open-source `Coliop` or `ZIMPL`[21]
- Interfaced with a varying set (few/many) of general-purpose solvers for large problem classes (MILP, MINLP, conic, . . .)
- AML is a separate language, typically interpreted (not efficient)
- Mostly "flat" languages (no OOP), modularity an issue
- Focus on "model once, solve once"; some offer some support for iterative procedures but clearly an afterthought
- Hide the complexities of the model/solution process to inexperienced users

[18] https://ampl.com, https://www.gams.com
[19] https://www.aimms.com/platform/aimms-development
[20] https://www.ibm.com/docs/en/icos/12.8.0.0?topic=opl-optimization-programming-language
[21] http://www.coliop.org, https://zimpl.zib.de

Modelling systems, and what they are for

- Modelling systems: libraries written in general-purpose languages providing similar functionalities to AML
- Often open-source: `FLOPCpp`, `COIN Rehearse` and `Gravity`[22] (`C++`), `PuLP` and `Pyomo`[23](`Python`), `JuMP` (`Julia`) and `YALMIP`[24] (`Matlab`)
- May not fully replicate AML constructs, sometimes more limited
- Solver interfacing and overhead lower with efficient languages (`C++`)
- Multiple models and iterative procedures more natural
- Can exploit OOP features of host language for better modularity
- Mostly focus on general-purpose solvers and "model once, solve once"
- Tailored for end-users, not algorithms developers

[22] https://github.com/coin-or/FlopCpp, https://github.com/coin-or/Gravity https://github.com/coin-or/Rehearse

[23] https://github.com/coin-or/pulp, http://www.pyomo.org

[24] https://github.com/jump-dev/JuMP.jl, https://yalmip.github.io

Decomposition / structure-aware solvers

- Some solvers provide decomposition capabilities:
  - `Cplex` does Benders', structure automatic or user hints
  - `SCIP`[10] does B&C&P (one-level D-W), pricing & reformulation up to the user (plugins)
  - `GCG`[10] extends `SCIP` with automatic and user-defined (one-level) D-W and recently also a generic (one-level) Benders' approach
  - `DDSIP`[25] and `PIPS`[26] implement D-W for two-stage stochastic programs
  - The `BaPCoD` B&C&P code has been used to develop `Coluna.jl`[27], doing one-level D-W and (alpha) Benders', multi-level planned
- Other solvers use structure in different ways: `BlockIP`[28], `OOPS`[29]

[25] https://github.com/RalfGollmer/ddsip
[26] https://github.com/Argonne-National-Laboratory/PIPS
[27] https://github.com/atoptima/Coluna.jl
[28] http://www-eio.upc.edu/~jcastro/BlockIP.html
[29] https://www.maths.ed.ac.uk/~gondzio/parallel/solver.html

Decomposition-aware modelling systems: are there any?

- In a word?

[30] https://www.maths.ed.ac.uk/ERGO/sml
[31] https://github.com/StructJuMP/StructJuMP.jl
[32] https://github.com/atoptima/BlockDecomposition.jl

Decomposition-aware modelling systems: are there any?

- In a word? Very few (that's two words . . .)
- `OOPS` is interfaced with `SML`[30], providing some parallel capabilities
- `PIPS` is interfaced with `StructJuMP`[31], using `BlockDecomposition`[32]

[30] https://www.maths.ed.ac.uk/ERGO/sml
[31] https://github.com/StructJuMP/StructJuMP.jl
[32] https://github.com/atoptima/BlockDecomposition.jl

Decomposition-aware modelling systems: are there any?

- In a word? Very few (that's two words . . .)
- `OOPS` is interfaced with `SML`[30], providing some parallel capabilities
- `PIPS` is interfaced with `StructJuMP`[31], using `BlockDecomposition`[32]
- No modelling system is focused on multi-level structure, non-general-purpose solvers, parallel, and modularity/extendability
- Although `JuMP` is doing a good job at promoting some of these

[30] https://www.maths.ed.ac.uk/ERGO/sml
[31] https://github.com/StructJuMP/StructJuMP.jl
[32] https://github.com/atoptima/BlockDecomposition.jl

Decomposition-aware modelling systems: are there any?

- In a word? Very few (that's two words . . .)
- `OOPS` is interfaced with `SML`[30], providing some parallel capabilities
- `PIPS` is interfaced with `StructJuMP`[31], using `BlockDecomposition`[32]
- No modelling system is focused on multi-level structure, non-general-purpose solvers, parallel, and modularity/extendability
- Although `JuMP` is doing a good job at promoting some of these
- We tried working with `Julia`, but most solvers are in `C` / `C++`, and the full circle `Julia` → `C++` → `Julia` did not work well
- So we choose no-performance-compromise `C++`, accepting the drawbacks

[30] https://www.maths.ed.ac.uk/ERGO/sml
[31] https://github.com/StructJuMP/StructJuMP.jl
[32] https://github.com/atoptima/BlockDecomposition.jl

Conclusions
(Part III)

Conclusions (Part III)

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

Conclusions (Part III)

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

Conclusions (Part III)

- 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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "plus some branching" to deal with nonconvexity

Conclusions (Part III)

- 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 60+-years old Cutting-Plane algorithm[3] "plus some branching" to deal with nonconvexity
- Different twists, different conditions to work:

Conclusions (Part III)

- 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 60+-years old Cutting-Plane algorithm[3] "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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "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 III)

- 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 60+-years old Cutting-Plane algorithm[3] "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
- Assume this is done for you (another story – course)