Decomposition in Large-Scale Optimization: Old Ideas and New Developments

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

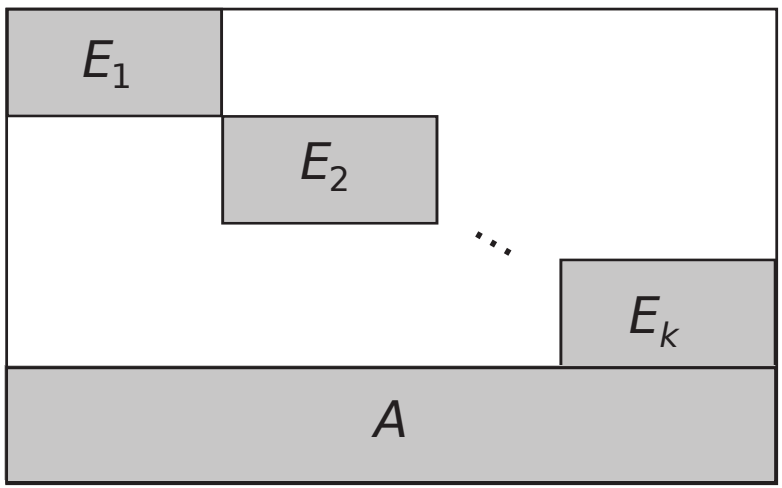
Cagliari

January 29 – 30, 2015

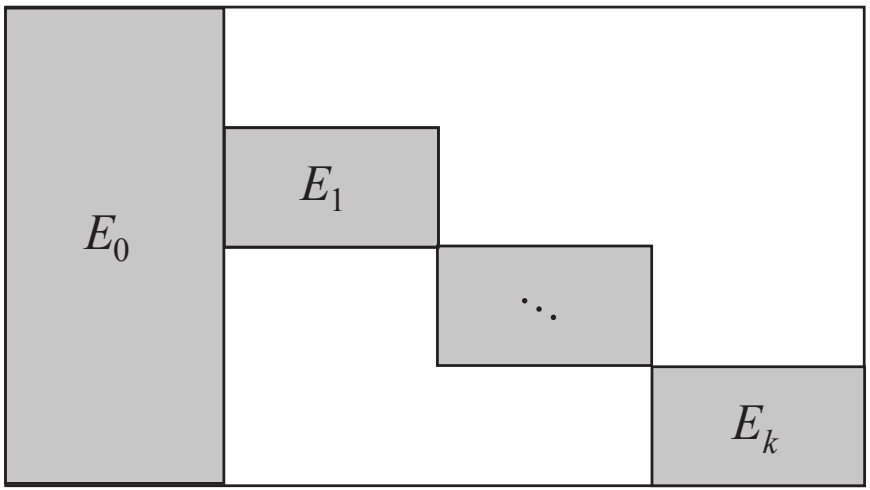
Outline

Block-Structured Programs

- (Challenging) applications of Integer Linear Programming are large-scale: millions of variables/constraints
- Good news: 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 (Integer) 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 y_s, different for each scenario $s \in S$, constraints $E_0^s x + E_s y_s \leq b_s$
- Minimize here-and-now cost plus average cost of reserve actions

$$\min \left\{ c_0 x + \sum_{s \in S} \pi_s c_s y_s \ : \ E_0 x \leq b_0 \ , \ E_0^s x + E_s y_s \leq b_s \quad s \in S \right\}$$

- Extends to multi-stage (structure repeats "fractally" into each E_s)
- Often other structures inside E, network a common one
- Extends to other risk measures (CVaR, ...), integer variables, ...
- Many applications: energy [0], water, logistics, telecom, finance, ...

---

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

(Very) Classical decomposition approaches I: primal

- Structure $\Longrightarrow$ use decomposition approaches
- Here-and-now decisions are naturally complicating
- Main idea: define the (expected, nonlinear) value function

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

  decomposes $\Longrightarrow$ "easy" to compute (but can be ∞)
- Construct Benders' reformulation [1]

$$\min \left\{ v(x) \ : \ E_0 x \leq b_0 \right\}$$

  a much smaller but nonlinear equivalent problem
- A "complicated" function that can be evaluated at each x
- Can use appropriate algorithms to solve it [2]

---

[1] Benders "Partitioning procedures for solving mixed-variables programming problems" *Numerische Mathematik* 1962

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

(Very) Classical decomposition approaches II: dual

- Alternative approach: introduce artificial complicating constraints

$$\begin{aligned} &\min c_0 x + \textstyle\sum_{s\in S} \pi_s c_s y_s && && (1)\\ &E_0 x \leq b_0 \\ &E_0^s x_s + E_s y_s \leq b_s && s \in S \\ &x_s = x && s \in S && (2) \end{aligned}$$

- Relax (2) into (1) with multipliers λ_s, Lagrangian function [3]

$$\begin{aligned} f(\lambda) = \min \quad & c_0 x + \textstyle\sum_{s\in S} \pi_s c_s y_s + \sum_{s\in S} \lambda_s (x - x_s) \\ & E_0 x \leq b_0 \\ & E_0^s x_s + E_s y_s \leq b_s && s \in S \end{aligned}$$

  decomposes $\Longrightarrow$ "easy" to compute

- The Lagrangian dual $\max\{ f(\lambda) \}$ equivalent to the problem (not necessarily much smaller, $|\lambda| = |x||S|$ may be $\approx |y|$)
- Can still be solved by [2]

---

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

Example II: Multicommodity Network Design

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

$$\min \sum_{k \in K} \sum_{(i,j) \in A} d^k c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{3}$$

$$\sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = \begin{cases} 1 & \text{if } i = s^k \\ 1 & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad i \in N\,,\ k \in K \tag{4}$$

$$\sum_{k \in K} d^k x_{ij}^k \leq u_{ij} y_{ij} \qquad (i,j) \in A \tag{5}$$

$$x_{ij}^k \in [0,1] \qquad (i,j) \in A\,,\ k \in K \tag{6}$$

$$y_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{7}$$

- $K \equiv$ commodities $\equiv (s^k\,,\ t^k\,,\ d^k)$ (not completely generic)
- Pervasive structure in most of combinatorial optimization
- Many applications: logistic, transportation, telecom, energy, ...
- Many sources of structure $\Longrightarrow$ the paradise of decomposition [4,5]

---

[4] Ford, Fulkerson "A Suggested Computation for Maximal Multicommodity Network Flows" *Management Science* 1958

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

Classical decomposition approaches

- Benders' decomposition [1] of linking variables:
  - design (y) variables are "naturally" linking if u_{ij} large
  - Benders' cuts are metric inequalities defining the multiflow feasibility
  - Linking variables can be artificially added if not present

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

  ("resource decomposition") [6]

- Lagrangian relaxation [3] of linking constraints:
  - (5): $\Longrightarrow$ flow (shortest path) relaxation (integrality property $\equiv$ "easy")
  - (4): $\Longrightarrow$ knapsack relaxation (only one integer variable per problem)
  - others possible

- Let's see how they work

---

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

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

Block-diagonal Linear Program

- Block-diagonal LP (linking constraints)

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

$$X = \bigotimes_{k \in K} X^k = \{ x^k \ : \ E^k x^k \leq d^k \}$$

- $|K|$ is large so, (Π) is very large

- We know how to efficiently optimize upon X, for two reasons:
  - a bunch of (many, much) smaller problems instead of a large one
  - the X^k may have structure (shortest path, knapsack, ...)

- We could efficiently solve (Π) if linking constraints removed: how to exploit it?

Dantzig-Wolfe reformulation

- Dantzig-Wolfe reformulation (temporarily assume X compact):
  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

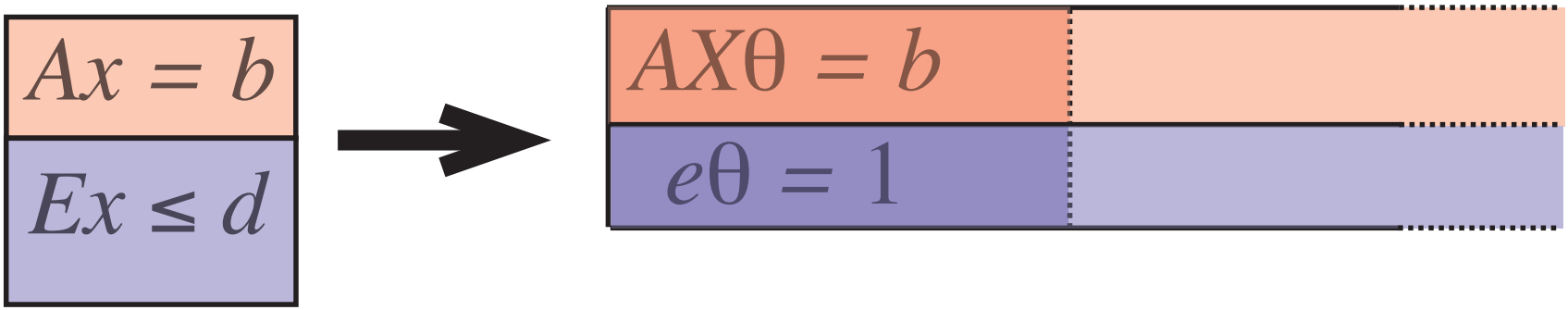
$$(\Pi) \qquad \left\{ \begin{array}{rrl} \max & c \left(\sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} \right) & \\ & A \left(\sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} \right) & = b \\ & \sum_{\bar{x} \in X} \theta_{\bar{x}} & = 1 \quad , \quad \theta_{\bar{x}} \geq 0 \qquad \bar{x} \in X \end{array}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

- How many points? Only the vertices $V \subset X$ of X are required
- 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

Dantzig-Wolfe decomposition ≡ Column Generation

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



  linear program with (exponentially) many columns

- But, efficiently optimize over $X \Longrightarrow$ generate vertices ($\equiv$ columns)
- $\mathcal{B} \subset X$ (small), solve restriction of (Π) with $X \to \mathcal{B}$, i.e.,

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

  "master problem" ($\mathcal{B}$ small, not too costly)

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

Dantzig-Wolfe decomposition ≡ Lagrangian relaxation

- How do I tell if $\mathcal{B}$ contains the "right" columns? Use duality
- "Abstract" view of the master problem:

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

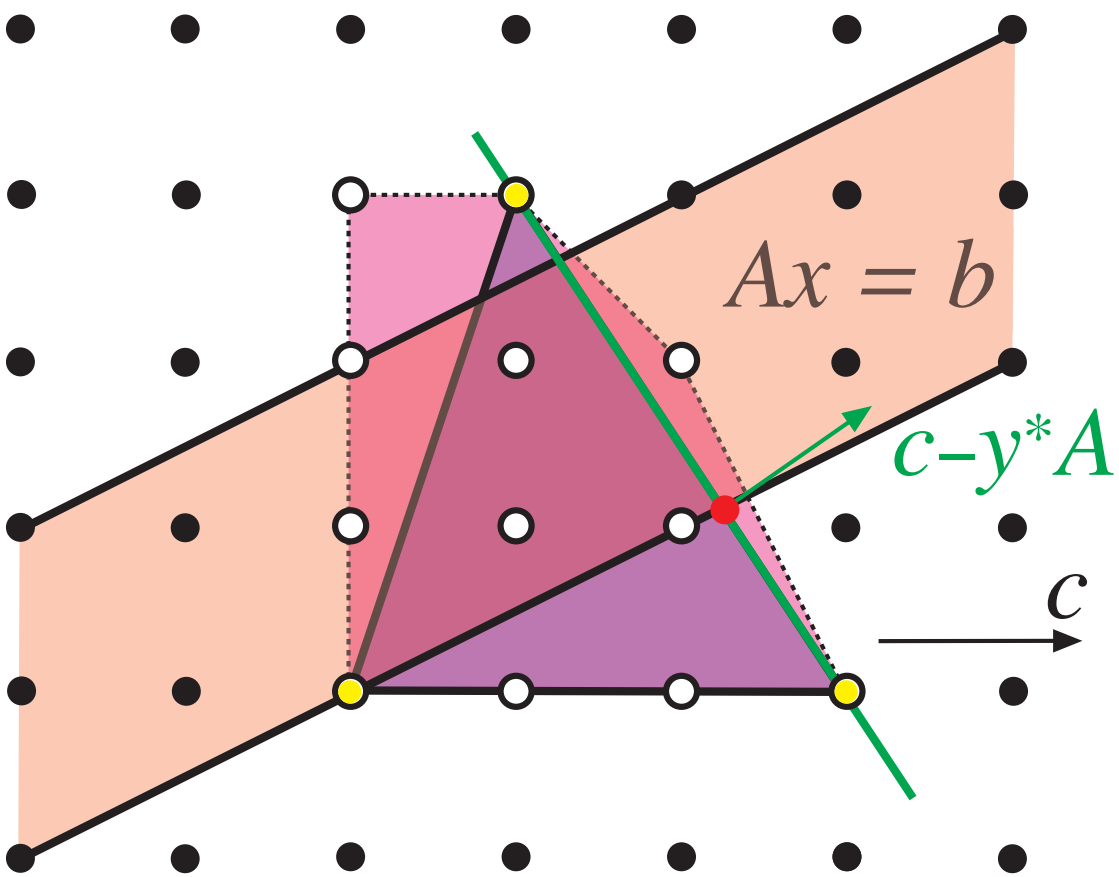
- Has a (linear) dual, (partial) dual optimal solution y^* of $Ax = b$
- Feed y^* to pricing problem (a.k.a. Lagrangian relaxation)

$$(\Pi_{y^*}) \qquad \max \{ (c - y^*A)x \ : \ x \in X \} \quad [+ y^*b]$$

(the whole of X, not $\mathcal{B}$, but we can do it efficiently)

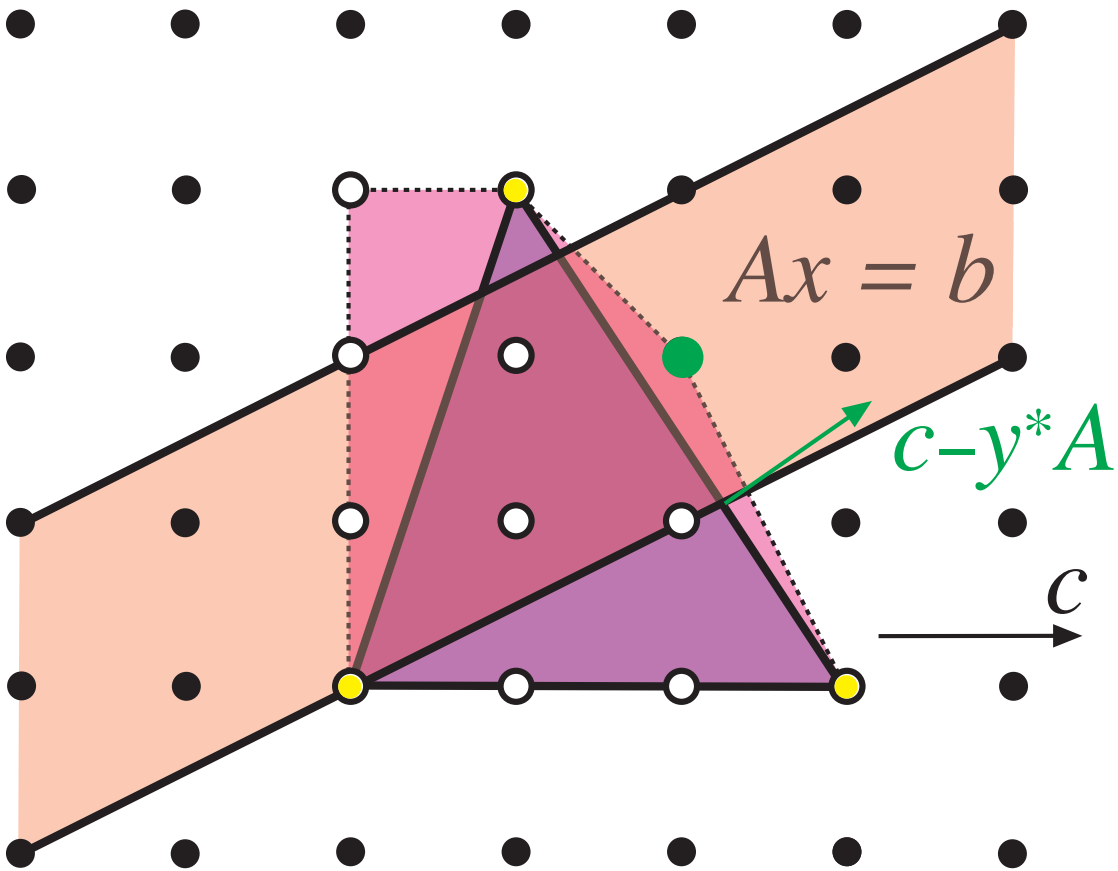
- If primal optimal solution $\bar{x}$ ($\equiv$ column) of (Π_{y^*}) has negative reduced cost $(c - y^*A)(x^* - \bar{x})$, use it to enlarge $\mathcal{B}$

Geometry of Dantzig-Wolfe/Column Generation



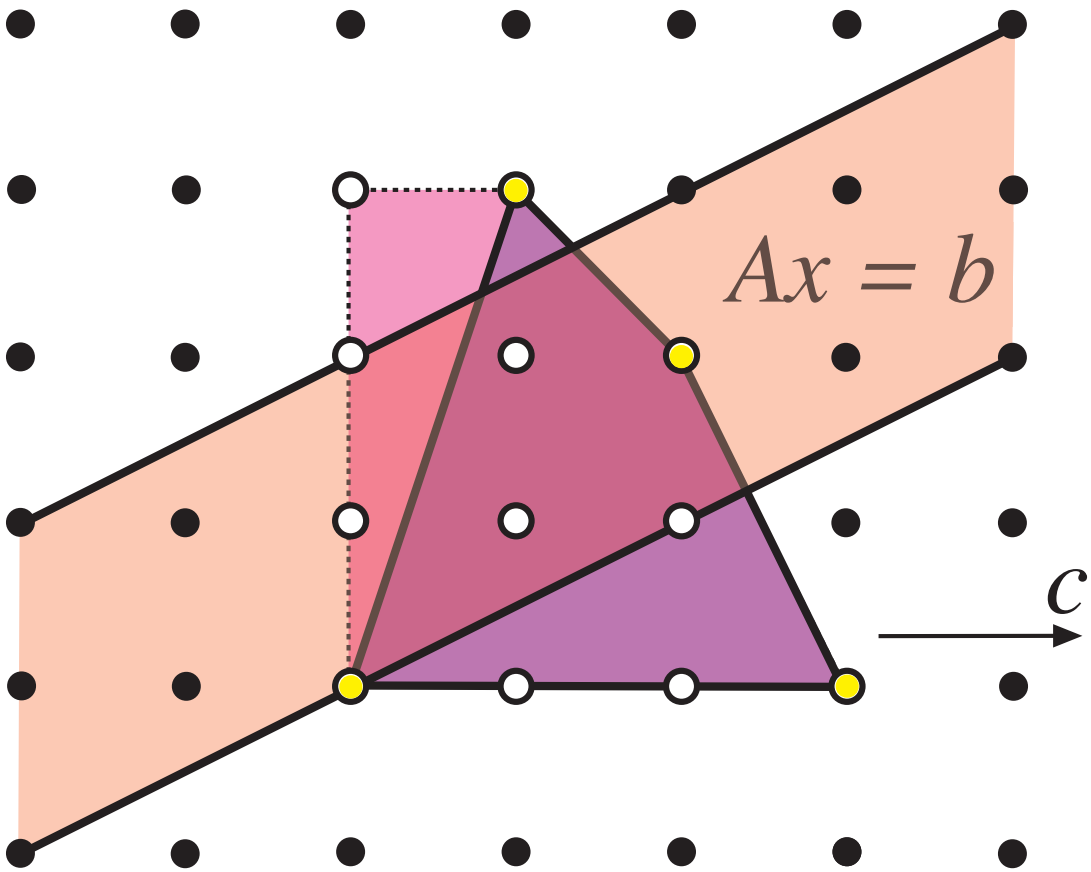
- $c - y^*A$ separates $conv(\mathcal{B}) \cap Ax = b$ from all $x \in X$ better than x^*

Geometry of Dantzig-Wolfe/Column Generation



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

Geometry of Dantzig-Wolfe/Column Generation



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

The Lagrangian dual

- Dual of $(\Pi_{\mathcal{B}})$:

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

  (note: $x \in \mathcal{B}$ "constraints index")

- $f_{\mathcal{B}}$ = lower approximation of "true" Lagrangian function

$$f(y) = \max \left\{ cx + y(b - Ax) \ : \ x \in X \right\}$$

  "easy" computability of $f(y)$ the only requirement

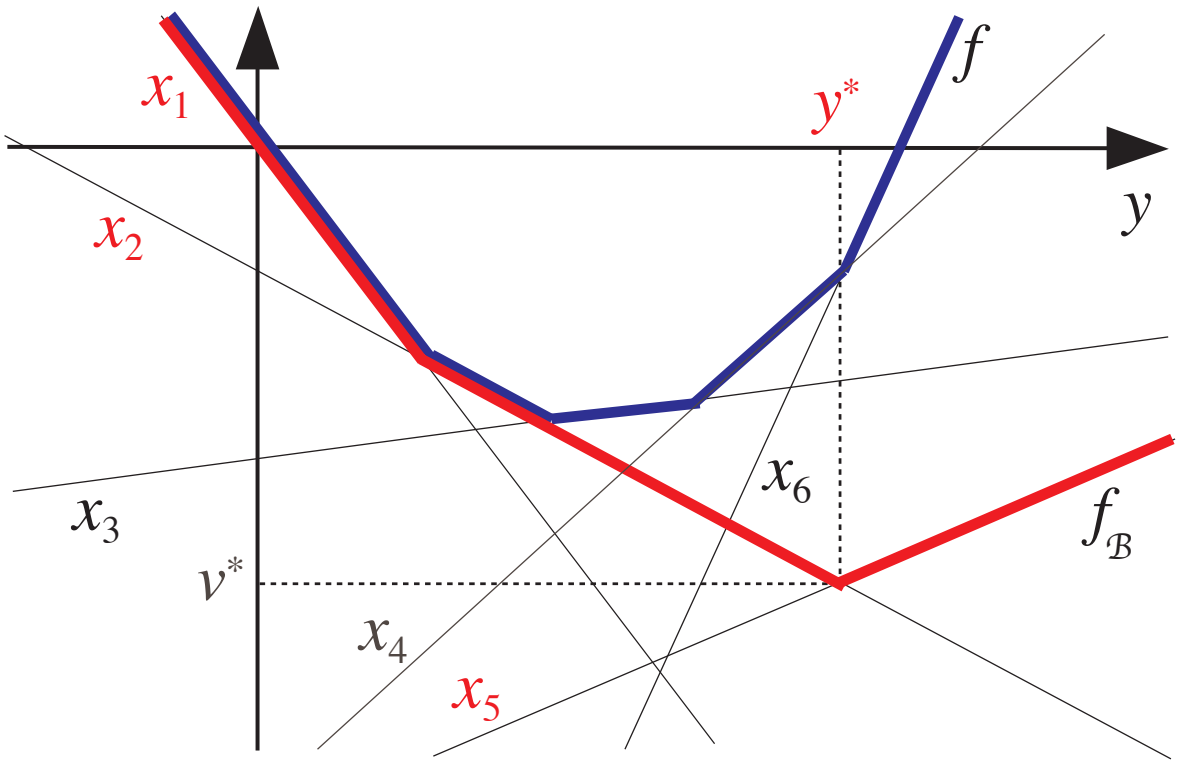
- Thus, $(\Delta_{\mathcal{B}})$ outer approximation of the Lagrangian dual

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

  that is equivalent to (Π)

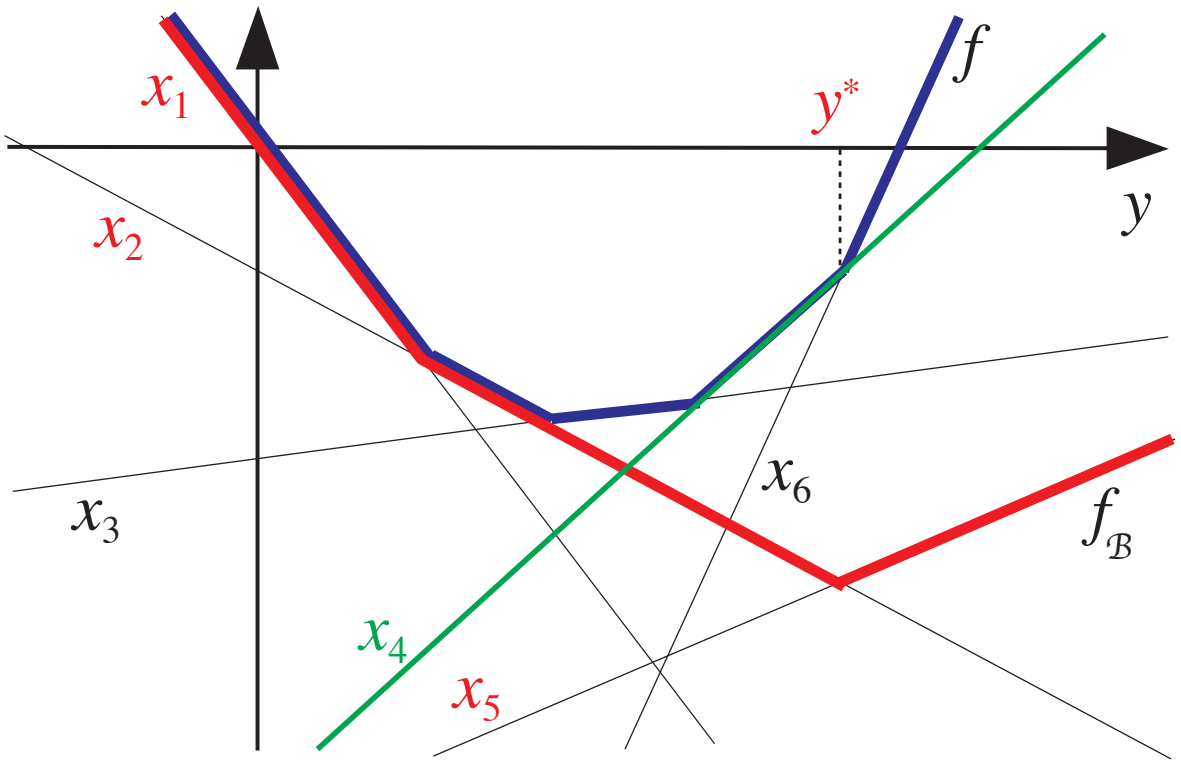
- Dantzig-Wolfe decomposition $\equiv$ Cutting Plane approach to (Δ) [2]

Geometry of the Lagrangian dual



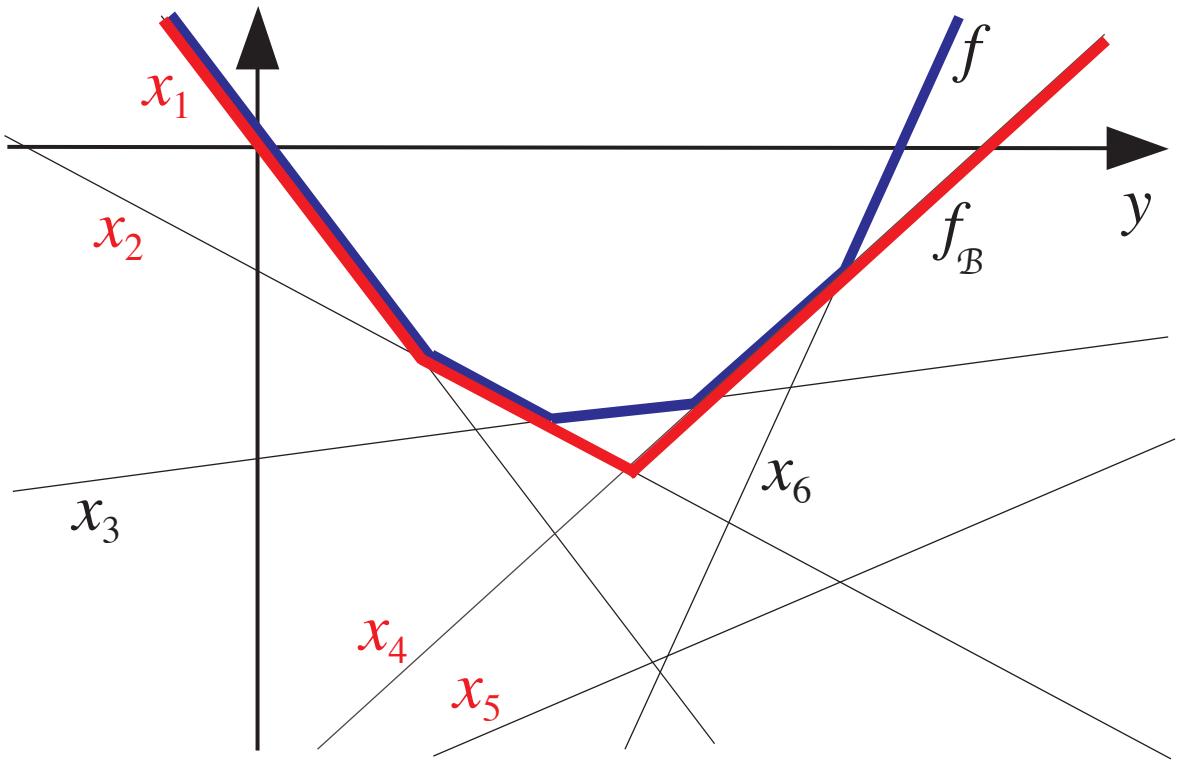
- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$

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^*)

The Integer Case: relationships with B&C

A Structured Integer Program

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

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

- If we can still efficiently optimize over X, due to size (decomposition) and/or structure (integrality), nothing changes
- What are we solving? Obviously, a (possibly tight) relaxation

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

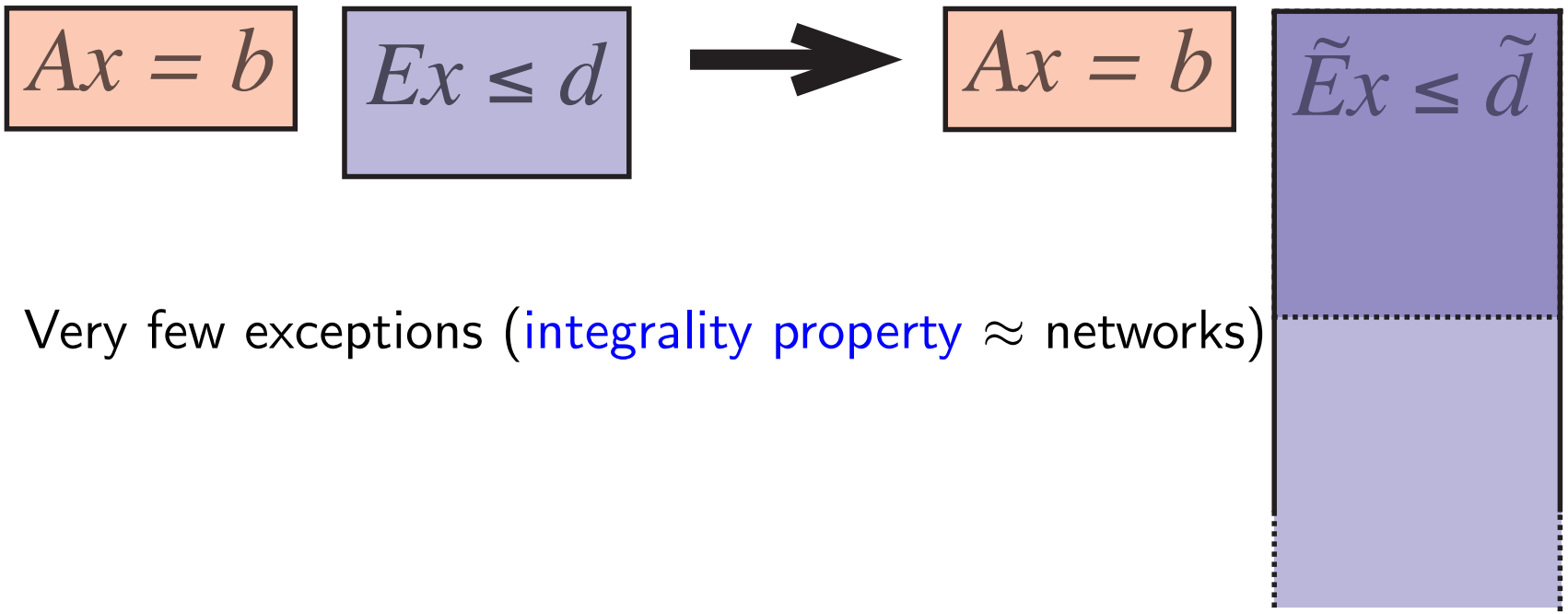
- Often does not solve $(\bar{\Pi})$, but gives (good) bounds
  $\Longrightarrow$ Branch & Bound with DW/Lagrangian/CG $\equiv$ Branch & Price
- Branching nontrivial: may destroy subproblem structure
  $\Longrightarrow$ branch on x (but $(\Pi_{\mathcal{B}})$ is on θ)
- Note: $\{ x \in \mathbb{R}^n : Ex \leq d \} = conv(X)$ (integrality) is bad
  $\Longrightarrow$ bound not better than standard linear relaxation

Alternative: a Good Formulation for X

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

Alternative: a Good Formulation for X

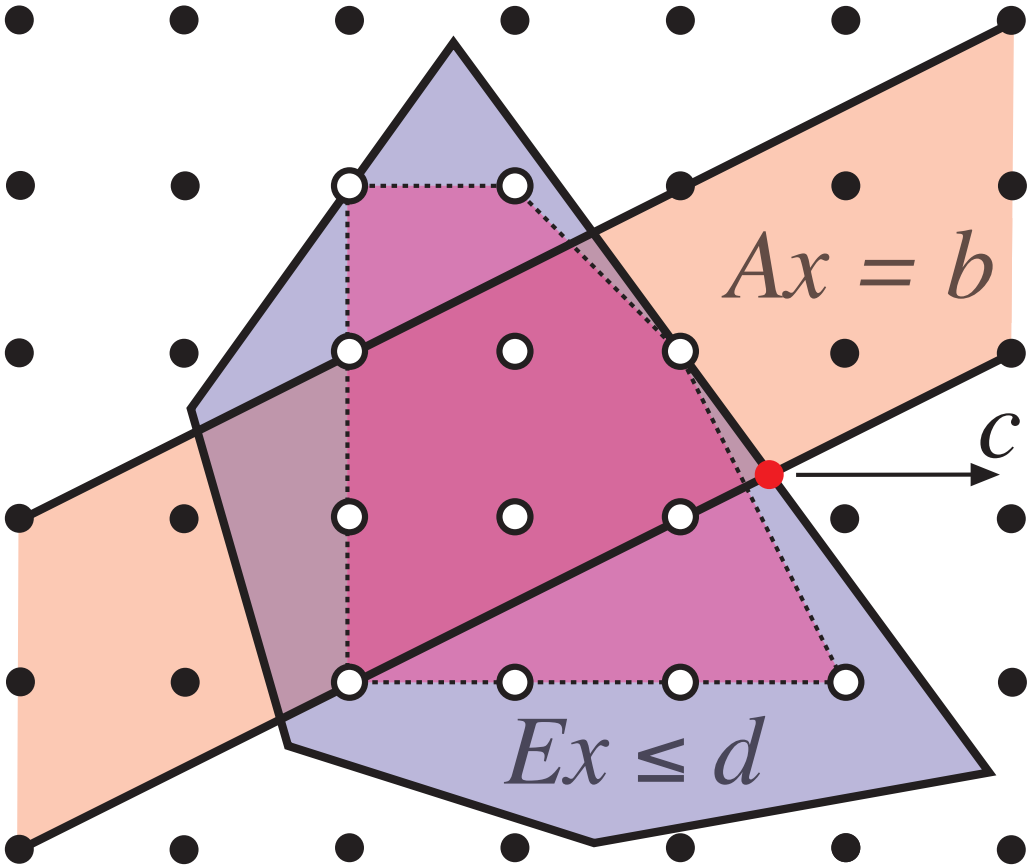
- (Under mild assumptions) $conv(X)$ is a polyhedron $\Longrightarrow$ $conv(X) = \{ x \in \mathbb{R}^n : \tilde{E}x \leq \tilde{d} \}$
- There are good formulations for the problem
- Except, practically all good formulations are too large



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

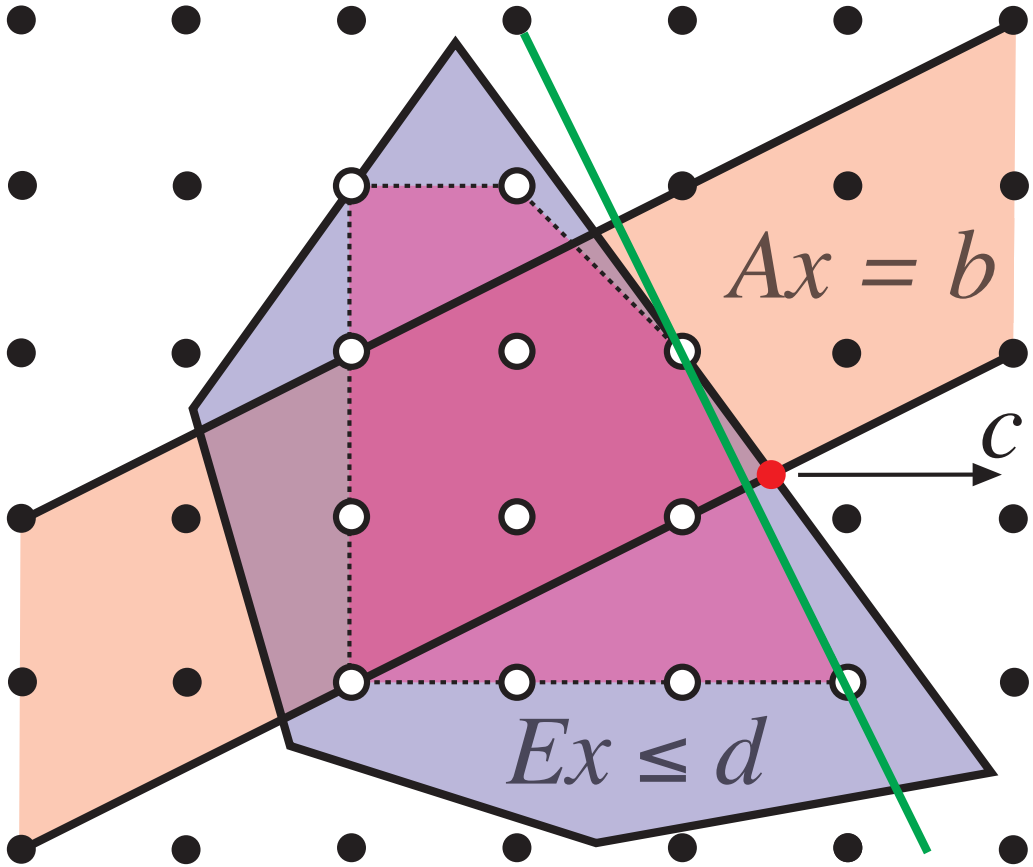
Row generation/polyhedral approaches

- The good news is: rows can be generated incrementally



Row generation/polyhedral approaches

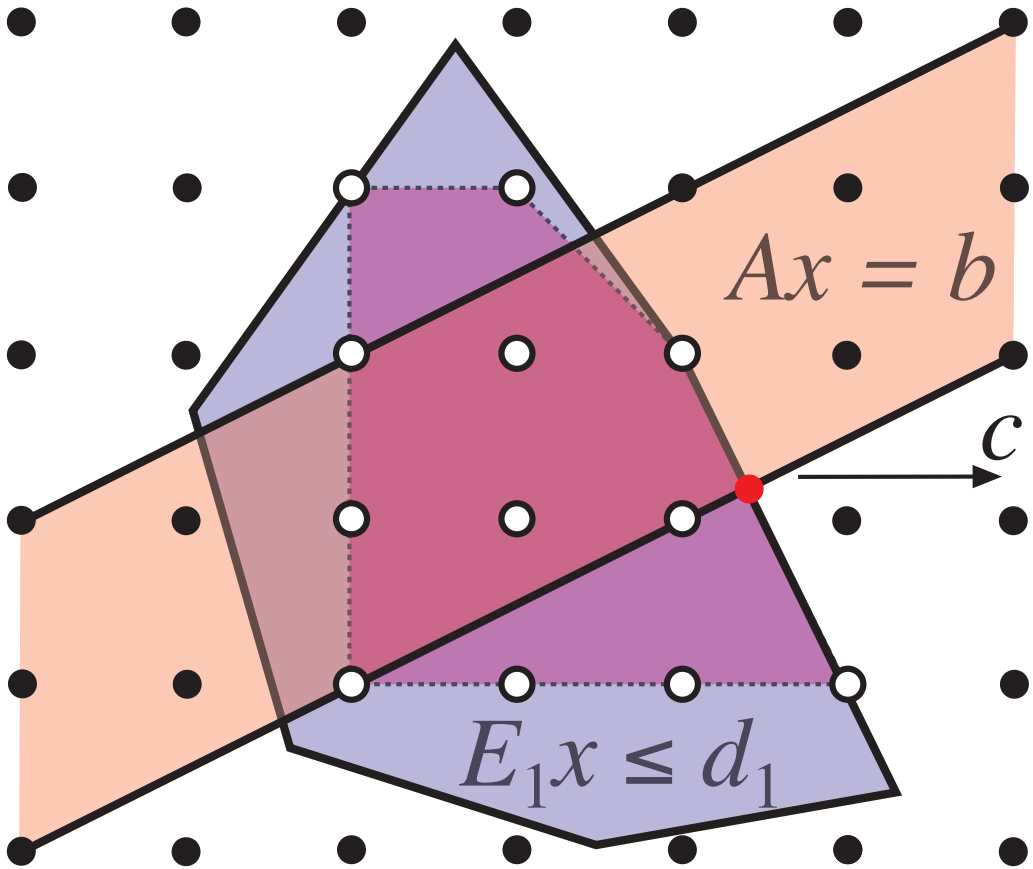
- The good news is: rows can be generated incrementally



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

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, a.k.a. Benders' decomposition Resource decomposition

Staircase-structured Linear Program

- Staircase-structured LP (linking variables)

$$(\Pi) \qquad \max \{ cx + ey \ : \ Ax + By \leq b \ , \ Ex \leq d \}$$

$$Ax + By \leq b \equiv A_k x + B_k y_k \leq b_k \quad k \in K$$

- $|K|$ is large so, (Π) is very large
- We know how to efficiently solve if x is fixed, for two reasons:
  - a bunch of (many, much) smaller problems instead of a large one
  - the B_k may have structure (shortest path, knapsack, ...)
- We could efficiently solve (Π) if linking variables fixed: how to exploit it?

Benders' reformulation

- Benders' reformulation: use value function

$$(B) \quad \max \left\{ cx + v(x) = \max\{ ey \ : \ By \leq b - Ax \} \ : \ Ex \leq d \right\}$$

then use duality to reformulate the inner problem

$$v(x) = \min \left\{ \lambda(b - Ax) \ : \ \lambda \in \Lambda = \{ \lambda \ : \ \lambda B = e \ , \ \lambda \geq 0 \} \right\}$$

- The polyhedron Λ does not depend on x, reformulate by points

$$(B) \quad \max \left\{ cx + v \ : \ v \leq \lambda(b - Ax) \ \ \lambda \in \Lambda \ , \ Ex \leq d \right\}$$

- How many points? Only the vertices $V \subset \Lambda$ of Λ are required
- Of course, in general the vertices are (exponentially) many
  but we can generate them solving the problem $v(x)$

Benders' decomposition

- Select (small) $\mathcal{B} \subset V$, solve master problem

$$
\begin{aligned}
(B_{\mathcal{B}}) \qquad & \max \left\{ cx + v \ : \ v \leq \lambda(b - Ax) \ \ \lambda \in \mathcal{B} \ , \ Ex \leq d \right\} \\
= \quad & \max \left\{ cx + v_{\mathcal{B}}(y) = \min\{ \lambda(b - Ax) \ : \ \lambda \in \mathcal{B} \} \ , \ Ex \leq d \right\}
\end{aligned}
$$

  (again, $\lambda \in \mathcal{B}$ "constraints index")

- $v_{\mathcal{B}} =$ lower approximation of "true" value function v
- Find (primal) optimal solution x^*, compute $v(x^*)$, rinse & repeat
- Benders' decomposition $\equiv$ Cutting Plane approach to (B) [2]
- Spookily similar to the Lagrangian dual, ain't it?
- Except, constraints are now attached to dual solutions $\lambda +$ $v(x) = \infty \Longrightarrow$ feasibility cut (extreme ray of Λ, details omitted)

Benders' decomposition for Integer Programs

- Staircase-structured ILP (integer linking vriables)

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

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

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

Conclusions (Part I)

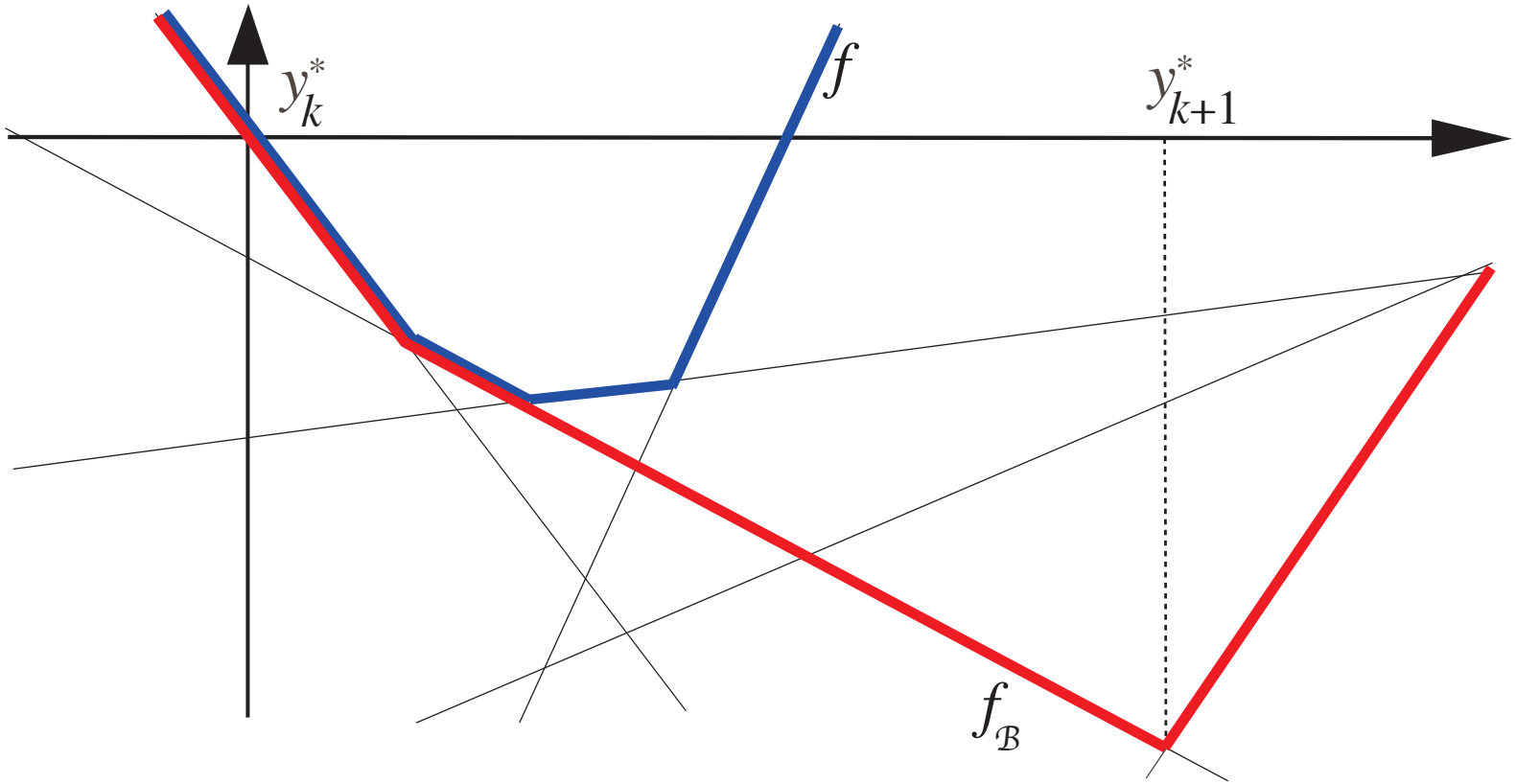
Conclusions (part I)

- Structured (Integer) Linear Programs are challening
- However, structure can be exploited by reformulation + duality
- Two different approaches, "primal" and "dual"
- Different twists, different conditions to work
  - who is complicating (constraints vs. variables), but tricks can be used to create the desired structure
  - who is reformulated (subproblem vs. master problem)
  - where integer variables are (subproblem vs. master problem)
  - where branching is done (subproblem vs. master problem)
- For linear programs, Lagrange is Benders' in the dual
- Both boil down to the Cutting Plane algorithm [2]
- 55 years old, does it work well? We'll see tomorrow

Stabilization

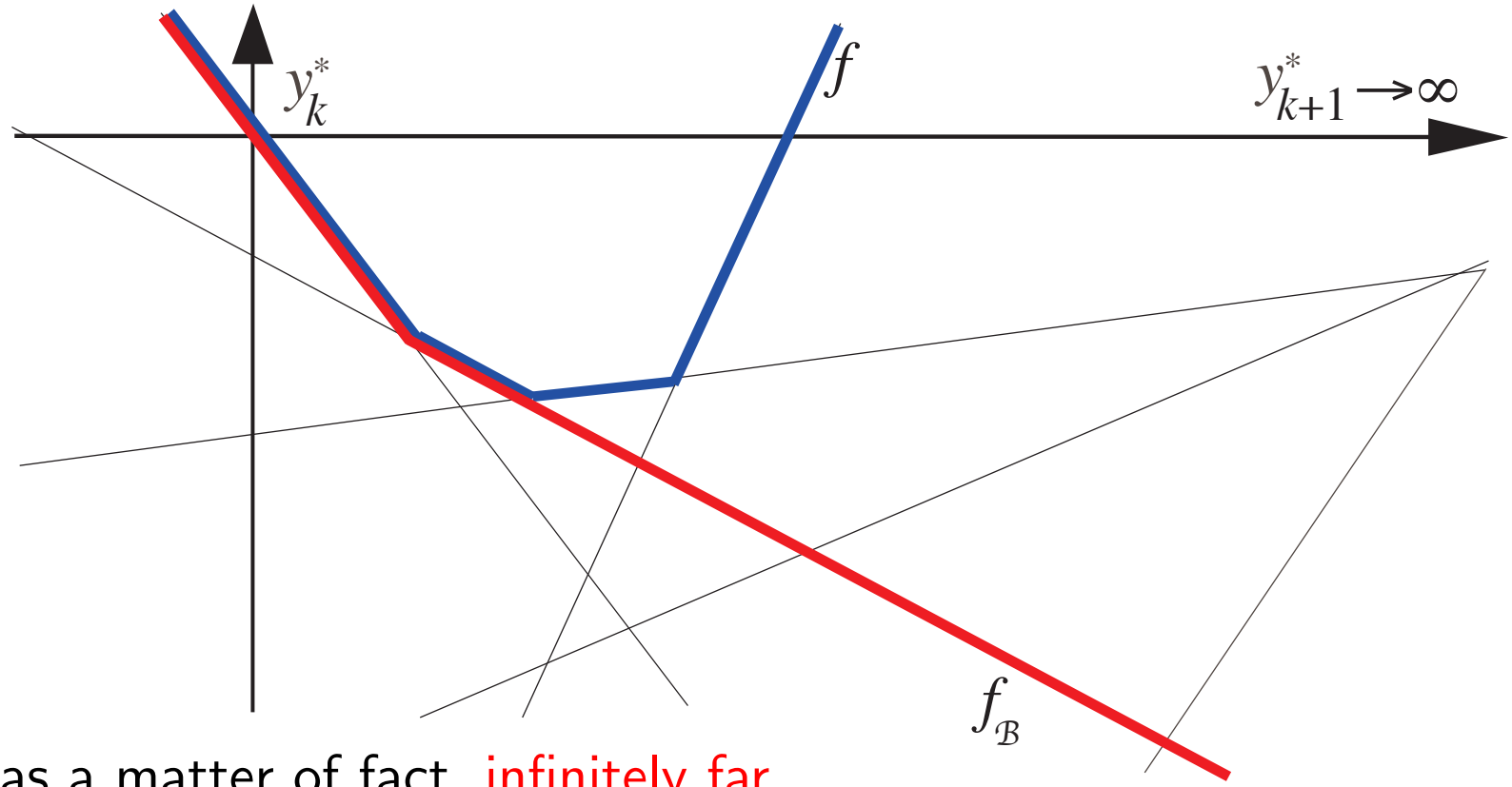
Issue with the Cutting Plane approach: instability

- y^*_{k+1} can be very far from y^*_k, where $f_{\mathcal{B}}$ is a "bad model" of f



Issue with the Cutting Plane approach: instability

- y^*_{k+1} can be very far from y^*_k, where $f_{\mathcal{B}}$ is a "bad model" of f



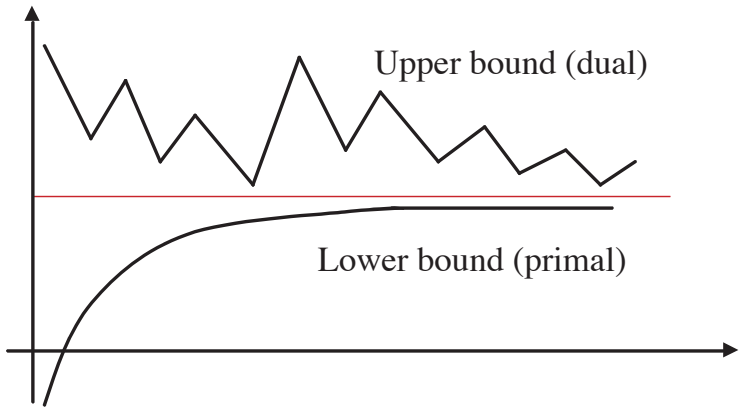
. . . as a matter of fact, infinitely far

- $(\Pi_{\mathcal{B}})$ empty $\equiv$ $(\Delta_{\mathcal{B}})$ unbounded $\Rightarrow$ Phase 0 / Phase 1 approach
- More in general: $\{ y^*_k \}$ is unstable, has no locality properties $\equiv$ convergence speed does not improve near the optimum

The effects of instability

- What does it mean?
  - a good (even perfect) estimate of dual optimum is useless!
  - frequent oscillations of dual values
  - "bad quality" of generated columns

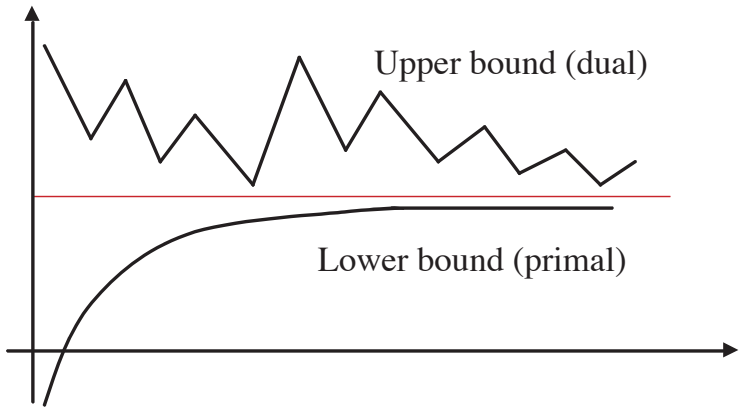
$\Longrightarrow$ tailing off, slow convergence



The effects of instability

- What does it mean?
  - a good (even perfect) estimate of dual optimum is useless!
  - frequent oscillations of dual values
  - "bad quality" of generated columns

$\Longrightarrow$ tailing off, slow convergence



- The solution is pretty obvious: stabilize it
- Gedankenexperiment: starting from known dual optimum, constrain duals in a box of given width

width	time		iter.		columns	
∞	4178.4	%	509	%	37579	%
200.0	835.5	20.0	119	23.4	9368	24.9
20.0	117.9	2.8	35	6.9	2789	7.4
2.0	52.0	1.2	20	3.9	1430	3.8
0.2	47.5	1.1	19	3.7	1333	3.5

Works wonders! ...

Stabilizing DW/Lagrange/CG

. . . if only we knew the dual optimum! (which we don't)

- Current point $\bar{y}$, box of size $t > 0$ around it
- Stabilized dual master problem [7]

$$(\Delta_{\mathcal{B},\bar{y},t}) \qquad \min \left\{ f_{\mathcal{B}}(\bar{y} + d) \ : \ \| d \|_\infty \leq t \right\} \tag{8}$$

- Corresponding stabilized primal master problem

$$(\Pi_{\mathcal{B},\bar{y},t}) \quad \max \left\{ cx + \bar{y}z - t\| z \|_1 \ : \ z = b - Ax \ , \ x \in conv(\mathcal{B}) \right\} \tag{9}$$

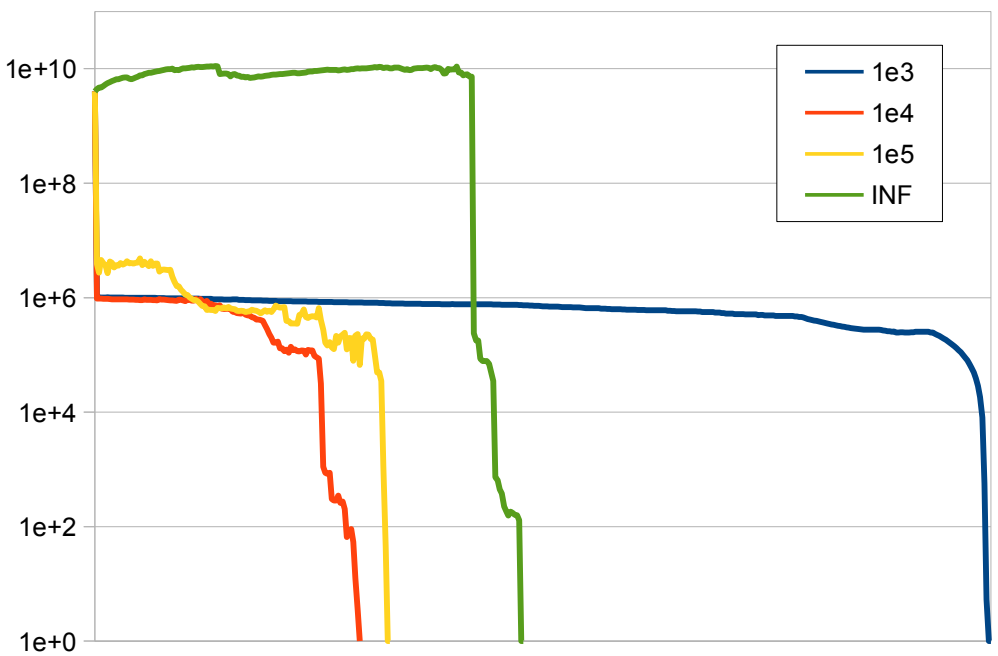
  i.e., just Dantzig-Wolfe with slacks

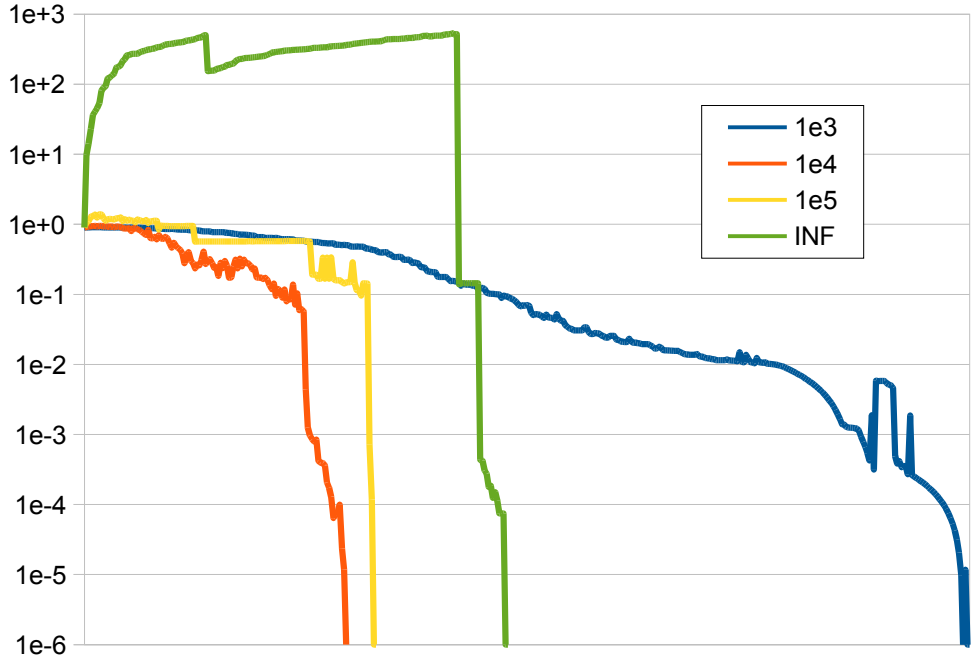
- When $f(\bar{y} + d^*) < f(\bar{y})$, move $\bar{y} = \bar{y} + d^*$ ("serious step")
- Uses just LP tools, relatively minor modifications
- How should one choose t?
- Does this really work?

---

[7] Marsten, Hogan, Blankenship "The Boxstep Method for Large-scale Optimization" *Operations Research* 1975

Computational results of the boxstep method (pds7)

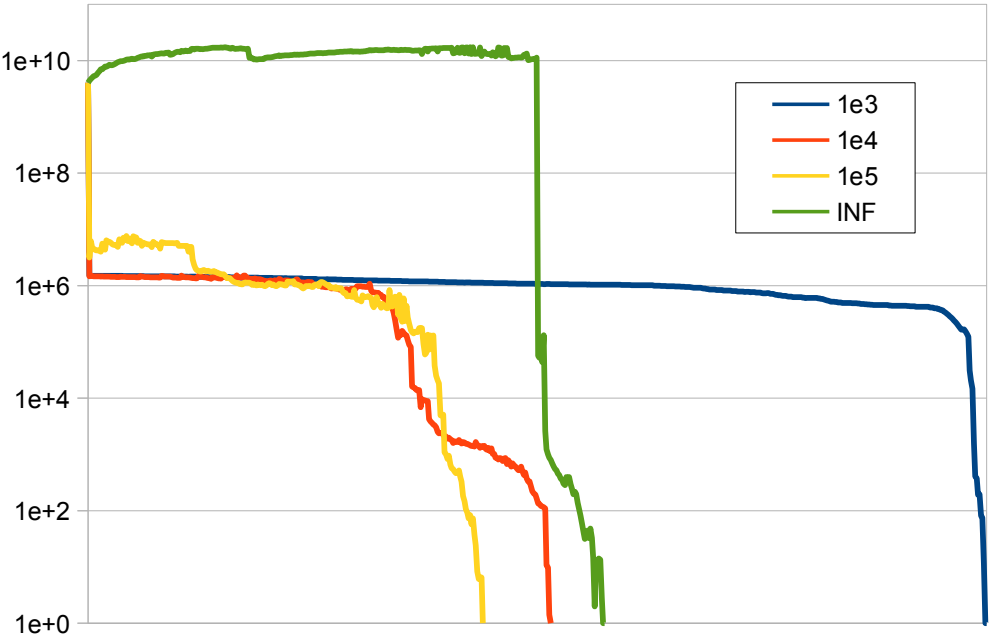


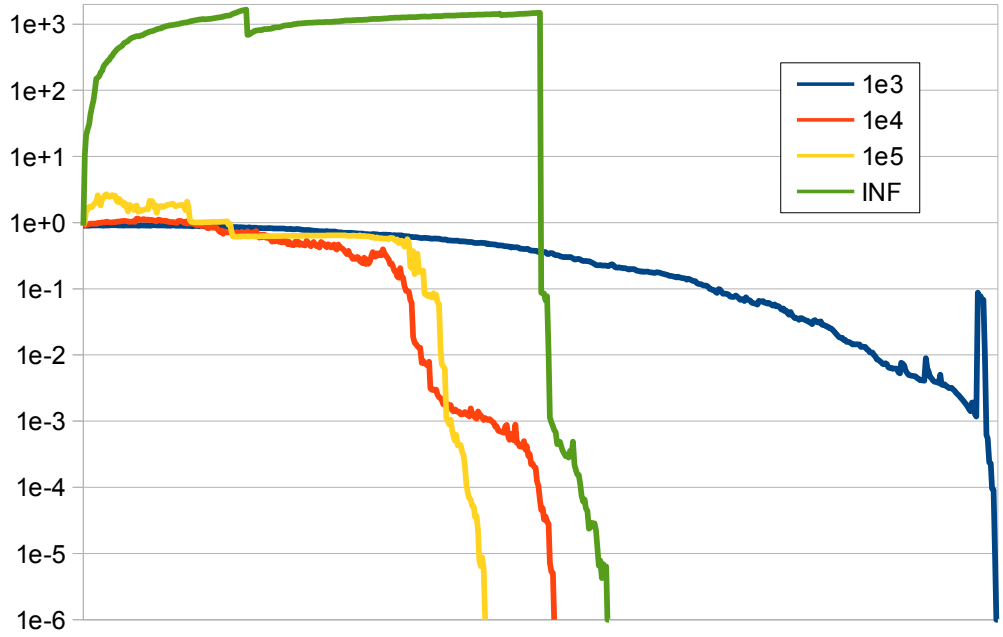


- Pure multicommodity flow instance (no y)
- Left = distance from final dual optimum

  right = relative gap with optimal value
- Stabilized with (fixed) different t, un-stabilized ($t = \infty$)
- One can clearly over-stabilize

Computational results of the boxstep method (pds18)



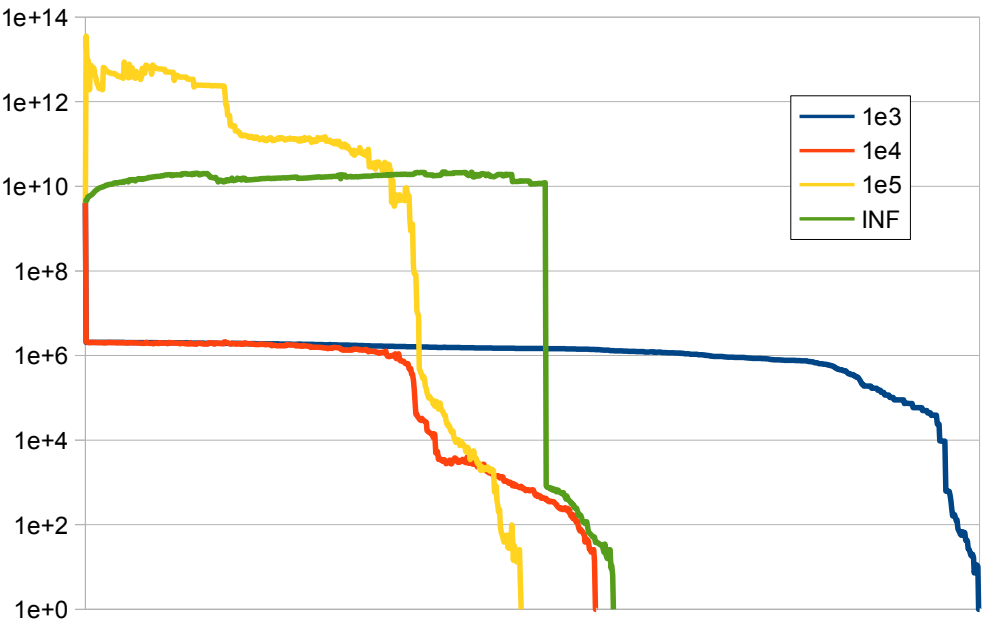


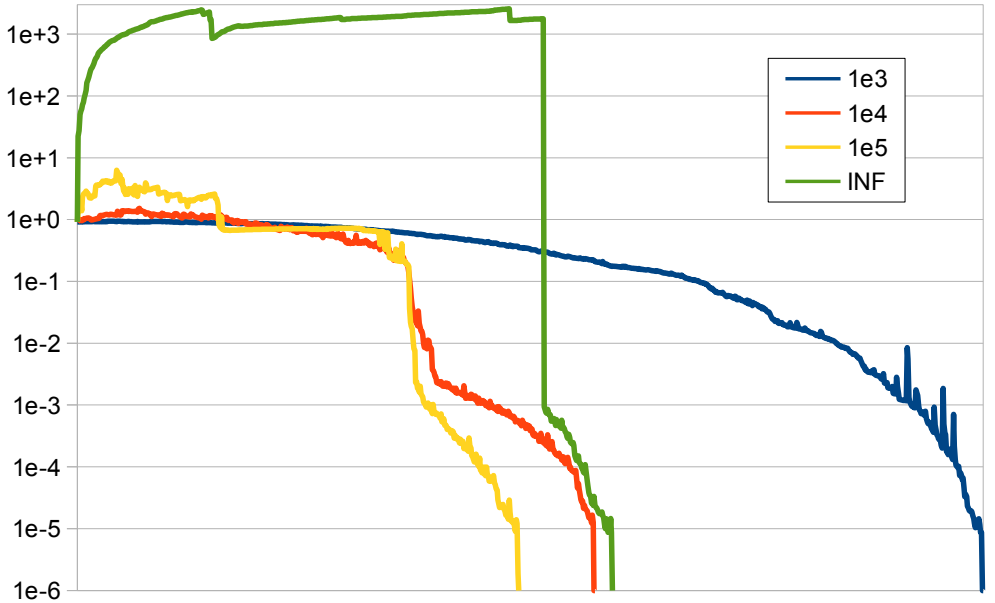
- All cases show a "combinatorial tail" where convergence is very quick
- $t = 1e3$: "smooth but slow" until the combinatorial tail kicks in a short-step approach not unlike subgradient methods [8]
- $t = \infty$: apparently trashing along until some magic threshold is hit
- "intermediate" t work best, but pattern not clear

---

[8] Camerini, Fratta, Maffioli "On Improving Relaxation Methods by Modified Gradient Techniques" *Math. Prog. Study* 1975

Computational results of the boxstep method (pds30)





- $t =$ 1e5: initially even worse than $t = \infty$ but ends up faster
- Clearly, some on-line tuning of t would be appropriate
- Perhaps a different stabilizing term would help? Why not [9]

$$(\Delta_{\mathcal{B},\bar{y},t}) \qquad \min \left\{ f_{\mathcal{B}}(\bar{y} + d) + \frac{1}{2t} \| d \|_2^2 \right\}$$

- "Because it's not LP" $\Longrightarrow$ a different duality need be used

---

[9] Lemaréchal "Bundle Methods in Nonsmooth Optimization" in *Nonsmooth Optimization* vol. 3, Pergamon Press, 1978

Generalized stabilization

- General stabilizing term $\mathcal{D}$, stabilized dual problem

$$(\Delta_{\bar{y},\mathcal{D}}) \qquad \phi_{\mathcal{D}}(\bar{y}) = \min\{ f(\bar{y}+d) + \mathcal{D}(d) \} \tag{10}$$

  with proper $\mathcal{D}$, $\phi_{\mathcal{D}}$ has same minima as f but is "smoother"

- Stabilized primal problem = Fenchel's dual of $(\Delta_{\bar{y},\mathcal{D}})$

$$(\Pi_{\bar{y},\mathcal{D}}) \qquad \min\{ f^*(z) - z\bar{y} + \mathcal{D}^*(-z) \} \tag{11}$$

  where $f^*(x) = \max_z\{ xz - f(z) \}$ the Fenchel's conjugate of f

- For our dual f, a generalized augmented Lagrangian

$$\max\{ cx + \bar{y}(b - Ax) - \mathcal{D}^*(Ax - b) \ : \ x \in conv(X) \} \tag{12}$$

- A "primal" exists even for a non-dual f: $v(\Pi) = -f^*(0) = v(\Delta)$ for

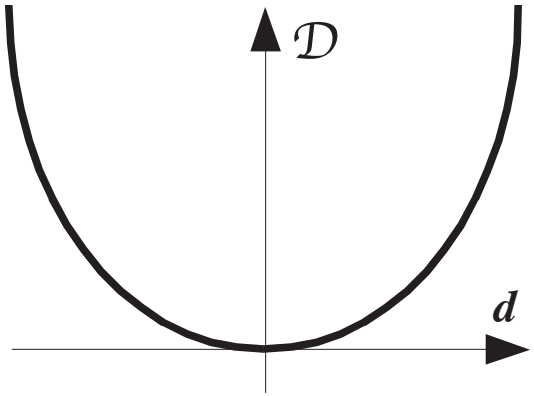
$$(\Pi) \qquad \max\{ -f^*(z) \ : \ z = 0 \}$$

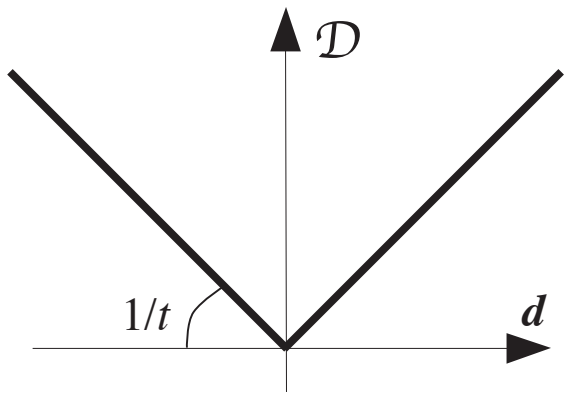
- General theory exist [10], but never mind

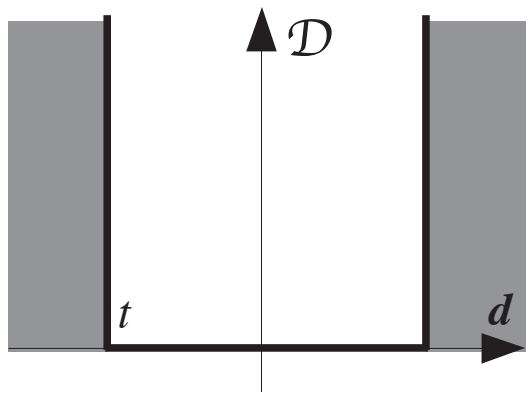
---

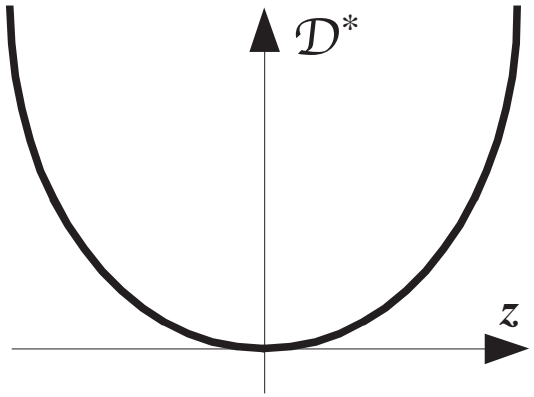
[10] F. "Generalized Bundle Methods" *SIAM Journal on Optimization* 2002

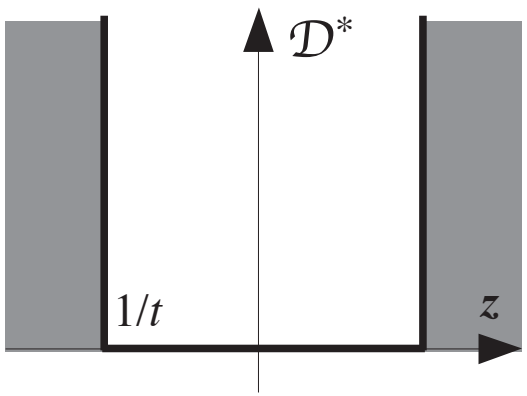
Classical stabilizing terms

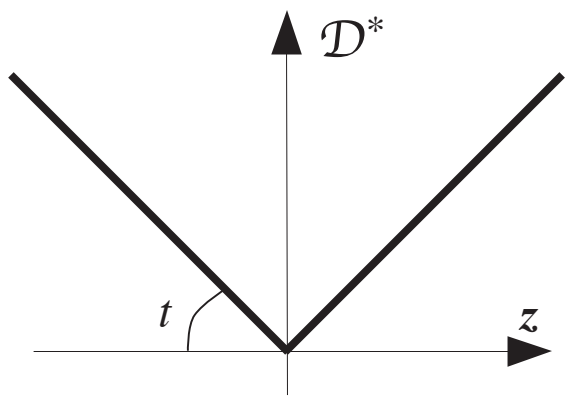










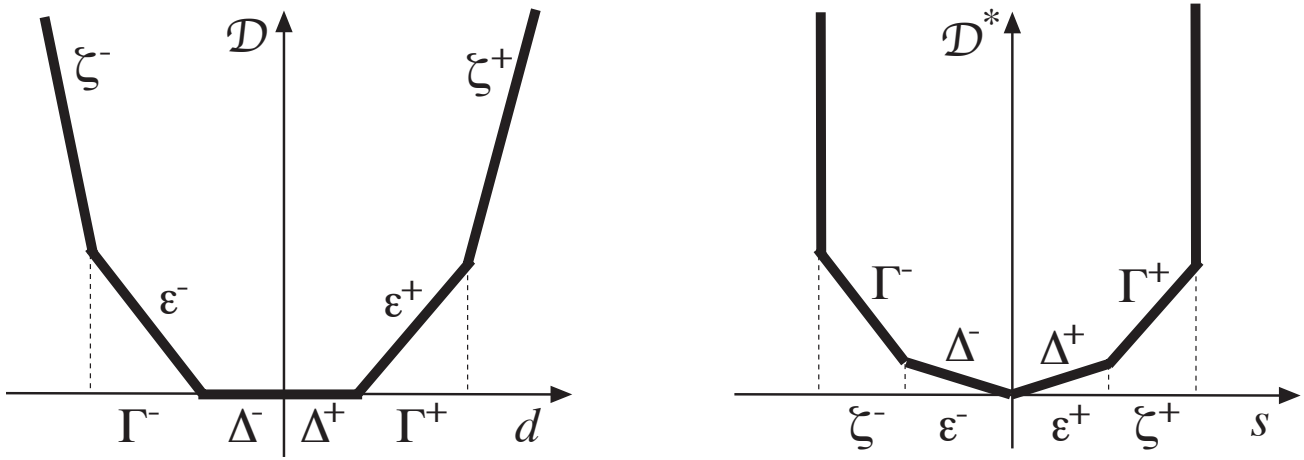


$$\mathcal{D} = \frac{1}{2t}\|\cdot\|_2^2 \qquad \mathcal{D} = \frac{1}{t}\|\cdot\|_1 \qquad \mathcal{D} = I_{B_\infty(t)}$$

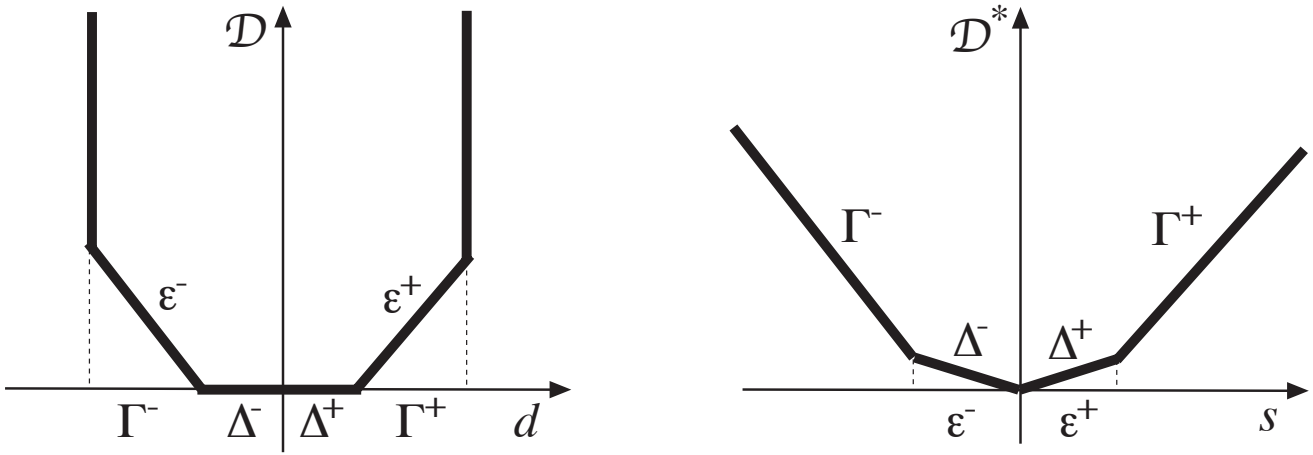
$$\mathcal{D}^* = \frac{1}{2}t\|\cdot\|_2^2 \qquad \mathcal{D}^* = I_{B_\infty(1/t)} \qquad \mathcal{D}^* = t\|\cdot\|_1$$

A 5-piecewise-linear function

Trust region on $\bar{y}$ + small penalty close + much larger penalty farther [11]



Slightly simplified version: only 3 pieces.



[11] Ben Amor, Desrosiers, F. "On the choice of explicit stabilizing terms in column generation" *Discrete Applied Math.* 2009

A 5-piecewise-linear master problem

$$
(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \begin{cases} \max & c\left(\sum_{\bar{x}\in\mathcal{B}} \bar{x}\theta_{\bar{x}}\right) - \bar{y}\left(s^- + w^- - w^+ - s^+\right) \\ & \qquad + \gamma^- s^- + \delta^- w^- + \delta^+ w^+ + \gamma^+ s^+ \\ & A\left(\sum_{\bar{x}\in\mathcal{B}} \bar{x}\theta_{\bar{x}}\right) + s^- + w^- - w^+ - s^+ = b \\ & \sum_{\bar{x}\in\mathcal{B}} \theta_{\bar{x}} = 1 \;,\; \theta_{\bar{x}} \geq 0 \quad \bar{x}\in\mathcal{B} \\ & 0 \leq s^- \leq \zeta^- \;,\; 0 \leq s^+ \leq \zeta^+ \\ & 0 \leq w^- \leq \varepsilon^- \;,\; 0 \leq w^+ \leq \varepsilon^+ \end{cases}
$$

A 5-piecewise-linear master problem

$$(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \begin{cases} \max \quad c\left(\sum_{\bar{x}\in\mathcal{B}} \bar{x}\theta_{\bar{x}}\right) - \bar{y}\left(s^- + w^- - w^+ - s^+\right) \\ \qquad\qquad + \gamma^- s^- + \delta^- w^- + \delta^+ w^+ + \gamma^+ s^+ \\ A\left(\sum_{\bar{x}\in\mathcal{B}} \bar{x}\theta_{\bar{x}}\right) + s^- + w^- - w^+ - s^+ = b \\ \sum_{\bar{x}\in\mathcal{B}} \theta_{\bar{x}} = 1 \;,\; \theta_{\bar{x}} \geq 0 \quad \bar{x}\in\mathcal{B} \\ 0 \leq s^- \leq \zeta^- \;,\; 0 \leq s^+ \leq \zeta^+ \\ 0 \leq w^- \leq \varepsilon^- \;,\; 0 \leq w^+ \leq \varepsilon^+ \end{cases}$$

- Same constraints as $(\Pi_{\mathcal{B}})$, 4 slack variables for each constraint
- Many parameters: *widths* $\Gamma^{\pm}$ and $\Delta^{\pm}$, *penalties* $\zeta^{\pm}$ and $\varepsilon^{\pm}$, different roles for small and large penalties
- Large penalties $\zeta^{\pm}$ easily make $(\Delta_{\mathcal{B},\bar{y},\mathcal{D}})$ bounded $\Longrightarrow$ no Phase 0
- 3-pieces: either large penalty $\Longrightarrow$ small moves, or small penalty $\Longrightarrow$ instability

Some computational results

- Comparing unstabilized, 5-piecewise and 3-piecewise penalty functions
- State-of-the-art `GenCol` code, large-scale, difficult MDVS instances (only root relaxation times)

		p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
time	CG	139	177	235	159	3138	3966	3704	1742	3685	3065
	PP-3	80	84	103	70	1173	819	1440	1143	1787	2283
	PP-5	33	36	38	28	482	335	946	572	1065	2037
iter	CG	117	149	200	165	408	524	296	186	246	247
	PP-3	82	92	104	75	181	129	134	145	144	189
	PP-5	47	47	49	45	93	64	98	83	86	150
mpt	CG	88	125	165	105	1679	2004	1955	925	1984	1743
	PP-3	44	47	60	42	572	399	740	543	858	1351
	PP-5	13	16	17	10	189	128	428	257	542	1326

- 5-pieces better than 3-pieces, 5-then-3 even better
- Quadratic more "stable", but optimized 5-pieces always better (quadratic has far less parameters, easier but less flexible)
- All this with fixed parameters, on-line adjustment possible (?)

On unboundedness and early termination

- A ray r of X: $x \in X \Longrightarrow x + \lambda r \in X$ for infinitely large λ
- $(c - yA)r > 0 \Longrightarrow f(y) = \infty \Longrightarrow$ constraint $cr \leq y(Ar)$ in the dual

$$(\Delta) \quad \min\{\, f(y) \,:\, y \in Y \,\}$$

  where facets of Y are dynamically generated like ordinary columns (constraint = column with a 0 in the convexity constraint)

- One might even hide the convexity constraint:
  - $A\bar{x} \rightarrow \left[\, A\bar{x} \,,\, 1 \,\right] \quad , \quad b \rightarrow \left[\, b \,,\, 1 \,\right]$;
  - Ignoring the special role of v (just another y)
  - Advantage: everything is a constraint

On unboundedness and early termination

- A ray r of X: $x \in X \Longrightarrow x + \lambda r \in X$ for infinitely large λ
- $(c - yA)r > 0 \Longrightarrow f(y) = \infty \Longrightarrow$ constraint $cr \leq y(Ar)$ in the dual

$$(\Delta) \quad \min\{\, f(y) \,:\, y \in Y \,\}$$

  where facets of Y are dynamically generated like ordinary columns (constraint = column with a 0 in the convexity constraint)

- One might even hide the convexity constraint:
  - $A\bar{x} \rightarrow \left[\, A\bar{x} \,,\, 1 \,\right] \quad , \quad b \rightarrow \left[\, b \,,\, 1 \,\right]$;
  - Ignoring the special role of v (just another y)
  - Advantage: everything is a constraint

This is a bad idea!

On unboundedness and early termination

- A ray r of X: $x \in X \Longrightarrow x + \lambda r \in X$ for infinitely large λ
- $(c - yA)r > 0 \Longrightarrow f(y) = \infty \Longrightarrow$ constraint $cr \leq y(Ar)$ in the dual

$$(\Delta) \quad \min\{\, f(y) \,:\, y \in Y \,\}$$

  where facets of Y are dynamically generated like ordinary columns (constraint = column with a 0 in the convexity constraint)
- One might even hide the convexity constraint:
  - $A\bar{x} \rightarrow \left[\, A\bar{x} \,,\, 1 \,\right] \quad , \quad b \rightarrow \left[\, b \,,\, 1 \,\right]$;
  - Ignoring the special role of v (just another y)
  - Advantage: everything is a constraint

This is a bad idea!

- Approximate stabilization = testing for decrease in f-value, but when a ray is generated, $f(\bar{y} + d^*) = +\infty$
- Convexity constraints are good: invent them if they are not there

Bundle vs. Proximal Point

- Same computational setting as before
- Comparing the same stabilization (5-piecewise) with (BP) or without (PP) early termination

		p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
time	CG	139	177	235	159	3138	3966	3704	1742	3685	3065
	PP	33	36	38	28	482	335	946	572	1065	2037
	BP	26	28	35	21	295	257	639	352	545	1505
iter	CG	117	149	200	165	408	524	296	186	246	247
	PP	47	47	49	45	93	64	98	83	86	150
	BP	37	43	44	36	57	53	59	49	51	101
mpt	CG	88	125	165	105	1679	2004	1955	925	1984	1743
	PP	13	16	17	10	189	128	428	257	542	1326
	BP	10	14	15	10	100	70	329	206	334	983

- Stabilization works well, approximate stabilization works better

Stabilization for Integer Master Problems (Benders')

- Stabilized master problem easy to do: with trust region

$$(B_{\mathcal{B},\bar{x},t}) \qquad \min \left\{ v_{\mathcal{B}}(x) \ : \ \| x - \bar{x} \|_\infty \leq t \ , \ x \in X \right\}$$

  pretty identical to (8) (no dual, though)

- For $X \subseteq \{0,1\}^n$, local branching constraint

$$\sum_{i \,:\, \bar{x}_i = 1} (1 - x_i) + \sum_{i \,:\, \bar{x}_i = 0} x_i \leq t$$

- However, when solved for one $\bar{x}$ only a local optima (nonconvex) $\Longrightarrow$ have to increase t until $t = n$ (∞)
- Silver lining: reverse box $\| x - \bar{x} \|_\infty \geq t$ (nonconvex) now easy
- Different idea possible: level stabilization [12]

$$(B_{\mathcal{B},\bar{x},l}) \qquad \min \left\{ \| x - \bar{x} \|_\infty \ : \ v_{\mathcal{B}}(x) \leq l \ , \ x \in X \right\} \qquad (13)$$

- Pros and cons: (13) can be solved inexactly, l somewhat easier to manage than t and need not go ∞, (13) larger (more difficult?), no reverse box, ... early days (no results to show)

---

[12] van Ackooij, F., de Oliveira "Inexact stabilized Benders decomposition approaches to chance-constrained problems with finite support" *working paper* 2015

Disaggregated Model

Dantzig-Wolfe and Multicommodity flows

- Dantzig-Wolfe reformulation (only flows, no y)
- $\mathcal{S} = \{$ (extreme) flows $s = [\bar{x}^{1,s}, \ldots, \bar{x}^{k,s}] \}$

$$\begin{array}{lll} \min & \sum_{s\in\mathcal{S}} \left(\sum_{k\in K} \sum_{(i,j)\in A} c_{ij}^k \bar{x}_{ij}^{k,s} \right) \theta_s & \\ & \sum_{s\in\mathcal{S}} \left(\sum_{k\in K} \bar{x}_{ij}^{k,s} - u_{ij} \right) \theta_s \leq 0 & (i,j) \in A \\ & \sum_{s\in\mathcal{S}} \theta_s = 1 \quad , \quad \theta_s \geq 0 & s \in \mathcal{S} \end{array}$$

- Another possibility: $X = X^1 \times X^2 \times \ldots \times X^{|K|} \Longrightarrow$ $conv(X) = conv(X^1) \times conv(X^2) \times \ldots \times conv(X^{|K|})$
- In practice: a different multiplier θ_s^k for each $\bar{x}^{k,s}$, with

$$\sum_{s\in\mathcal{S}} \theta_s^k = 1 \qquad k \in K$$

(clearly, previous case is $\theta_s^k = \theta_s^h, h \neq k$)

- Use more (convexity) constraints in the master problem

Disaggregated Multicommodity flows

- Simple scaling leads to arc-path formulation:
  $p \in \mathcal{P}^k = \{ s^k\text{–}t^k \text{ paths} \}$, c_p cost, $f_p (= d^k \theta^k_s)$ flow, $\mathcal{P} = \cup_{k \in K} \mathcal{P}^k$

$$
\begin{array}{lll}
\min & \sum_{p \in \mathcal{P}} c_p f_p & \\
& \sum_{p \in \mathcal{P} \,:\, (i,j) \in p} f_p \leq u_{ij} & (i,j) \in A \\
& \sum_{p \in \mathcal{P}^k} f_p = d^k & k \in K \\
& f_p \geq 0 & p \in \mathcal{P}
\end{array}
$$

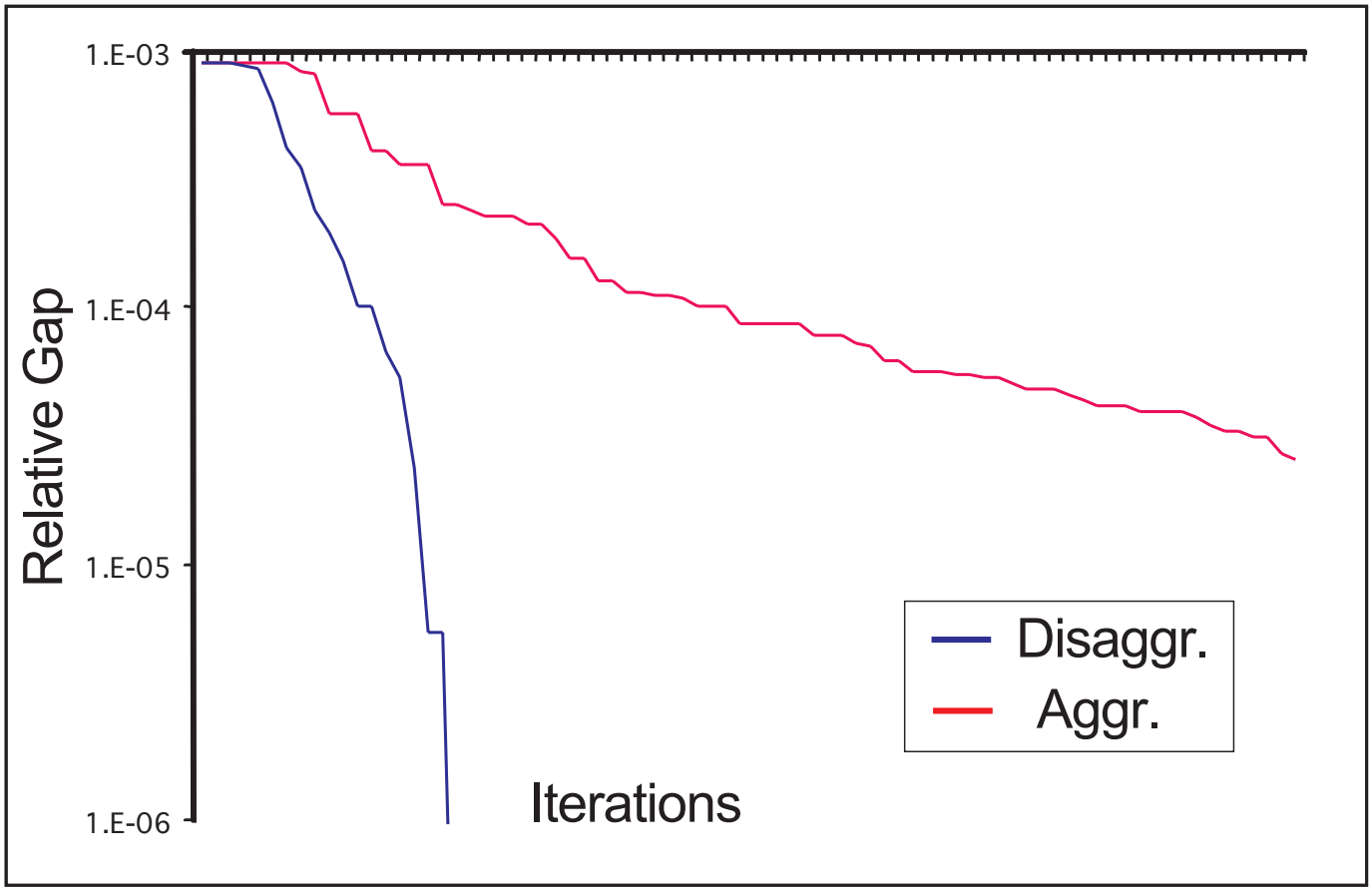
- Disaggregated formulation: more columns but sparser, more rows
- Much more efficient than aggregated formulation [13]
- Master problem size $\approx$ time increases, but convergence speed increases a lot $\equiv$ consistent improvement

---

[13] Jones, Lustig, Farwolden, Powell "Multicommodity Network Flows: The Impact of Formulation on Decomposition" *Mathematical Programming* 1993

Disaggregated decomposition

- Easily extended to any decomposable X [14]
- Stabilized versions immediate



[14] Borghetti, F., Lacalandra, Nucci "Lagrangian Heuristics Based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment" *IEEE Transactions on Power Systems* 2003

More Disaggregated Versions

- That was true 20 years ago with $|K| \approx 10$, still true if $|K| \approx 10000$?
- Aggregation is arbitrary, then why "all or nothing"?
- Partition $C = (C_1, C_2, \ldots, C_h)$ of K, partially aggregated model $f_{\mathcal{B}}^{C}$ with h components $f_{\mathcal{B}}^{i}$, each the sum over one C_i
- Basically, $\theta_s^k = \theta_s^h$ only for each $(h, k) \in C_i \times C_i$
- Exploring the trade-off between master problem size $\Longrightarrow$ time and iterations, subproblem time remains the same
- Aggregation index $\eta \in [0, 1]$:

$$h = |C| = \max\left\{\left\lceil (1 - \eta)|K| \right\rceil, 1\right\}$$

  $0 =$ fully disaggregated, $1 =$ fully aggregated

- How to choose the commodities in each C_i? In general open problem, here just group commodities with "close original names"

Even More Disaggregated Versions

- But what is a commodity, anyway?
  - Modeler's view: a product, origin-destination, stream of packets, ...
  - Algorithm's view: all that can be bunched together
- Commodity-independent data $\equiv$ bunch commodities with same origin
- Why is that? Because you can solve a unique SPT for all of them (which is because SPT has a funny auto-separability property)
- From a modeling viewpoint, there is no reason to (can always recover the original solution, less variables)
- This impact how the master problem is formulated [14] ...

Even More Disaggregated Versions

- But what is a commodity, anyway?
  - Modeler's view: a product, origin-destination, stream of packets, ...
  - Algorithm's view: all that can be bunched together
- Commodity-independent data $\equiv$ bunch commodities with same origin
- Why is that? Because you can solve a unique SPT for all of them (which is because SPT has a funny auto-separability property)
- From a modeling viewpoint, there is no reason to (can always recover the original solution, less variables)
- This impact how the master problem is formulated [14] ...
  or not: the Master Problem can be freely reformulated
- Aggregation index $\eta \in [-1, 0]$: $\overline{K}$ the number of OD pairs,
$$h = |C| = \max\left\{ \left\lceil -\eta|\overline{K}| \right\rceil, |K| \right\}$$
  $-1 =$ ODP formulation, $0 =$ DSP formulation [14]
- Again, commodities in a C_i just have "close destination node names"

(Preliminary, $\eta \geq 0$) Computational Results

- Generalized Bundle code using $D_t^* = \| \cdot \|_1$ (boxstep)
- Latest Cplex as Master Problem Solver
- Efficient implementation: overhead due to subgradient handling significant
- Limited effect of stabilization (not much need)
- (Reasonably) efficient subproblem solution with `MCFClass`

  `http://www.di.unipi.it/optimize/Software/MCF.html`

- Many instances, some old, some new, from

  `http://www.di.unipi.it/optimize/Data/MMCF.html`

- Results for $\eta < 0$ still brewing, but these significant enough already

Computational Results: Planar & Grid Instances

	0		0.2		0.4		0.6		0.8		1	
	time	it.	time	it.	time	it.	time	it.	time	it.	time	it.
grid7	2.5	12	2.91	13	2.39	15	2.12	14	2.62	18	285.76	1169
grid8	18.52	18	18.33	19	21.05	20	25.61	23	42.36	33	***	3848
grid9	36.04	15	45.94	16	60.54	18	85.99	20	189.92	32	***	2862
grid10	54.51	15	61.40	16	77.96	17	104.18	18	233.07	24	***	3848
grid12	61.64	11	61.24	10	65.44	11	71.81	11	148.89	13	***	2862
grid14	433.64	11	388.76	11	289.13	12	230.66	11	259.22	12	***	2862
planar100	2.16	14	1.96	13	1.42	13	2.36	13	2.74	15	25.49	1400
planar150	25.75	17	29.11	17	28.77	17	30.44	19	35.49	23	***	68896
planar300	21.34	13	22.86	14	23.54	14	24.12	15	24.71	14	1292.09	2967
planar500	15.27	11	14.75	11	13.91	11	12.71	12	10.84	11	197.62	317

- Large, nasty instances (you'll see later)
- *** = out of time limit (6400 seconds): all for $\eta = 1$, clearly worst
- Results somewhat erratic, but clearly $\eta = 0$ not always best

Computational Results: Goto & Mnetgen Instances

	0		0.2		0.4		0.6		0.8		1	
	time	it.	time	it.	time	it.	time	it.	time	it.	time	it.
Goto6-100	1.05	25	1.33	30	1.39	35	1.67	44	1.40	69	16.09	926
Goto6-400	1.45	15	1.59	17	1.76	19	2.40	22	5.79	32	60.53	1272
Goto6-800	2.41	12	2.54	14	2.85	15	3.62	17	9.24	25	134.42	1709
Goto8-10	2.96	75	4.57	104	6.14	137	7.68	164	18.12	301	45.29	722
Goto8-100	3.43	21	4.86	27	4.98	31	5.58	45	13.73	79	388.32	2114
Goto8-400	5.88	16	8.13	18	11.03	20	14.68	23	24.86	36	582.66	2690
Goto8-800	3.12	11	3.30	12	4.53	13	6.34	15	10.32	20	82.93	729
128-32	17.66	57	27.64	76	23.54	91	31.09	128	32.92	222	294.03	2753
128-32	57.23	46	66.04	59	63.66	70	79.97	92	108.53	169	1337.79	5296
128-64	95.45	34	125.27	43	126.71	50	147.25	65	174.81	108	1750.57	3741
128-128	5.68	109	5.73	109	8.08	158	12.34	209	24.09	437	25.22	449
256-8	31.65	140	45.55	183	77.50	252	94.51	276	289.69	635	1020.79	1826
256-16	146.37	148	181.38	219	244.79	271	404.15	381	885.73	704	1856.84	2175
256-32	400.59	117	510.74	163	640.14	200	1081.34	299	1666.35	480	2740.50	2615
256-64	563.66	86	744.93	113	1108.17	143	1624.06	196	1834.86	293	2670.98	1821

- . . . although in some cases $\eta = 0$ can be (almost) uniformly best

Computational Results: Waxman & Rmnet Instances

	0		0.2		0.4		0.6		0.8		1	
	time	it.	time	it.	time	it.	time	it.	time	it.	time	it.
W-50	1.43	3	0.17	3	0.11	3	0.07	3	0.03	3	0.04	9
W-100-6	1.53	2	0.20	2	0.13	2	0.09	2	0.04	2	0.06	10
W-100-10	1.34	3	0.38	3	0.32	3	0.27	3	0.22	3	0.70	15
W-100	1.50	2	2.10	2	1.37	2	0.98	2	0.72	2	1.06	7
W-150-6	2.44	2	2.30	2	1.81	2	1.20	2	0.63	2	4.54	44
W-150-10	1.23	3	0.83	3	0.66	3	0.60	3	0.14	2	0.48	4
W-150	3.23	3	4.74	3	3.17	3	2.70	3	0.67	3	4.49	9
4-8-11-100	0.56	5	1.31	5	0.83	5	0.58	5	0.40	5	0.31	8
4-8-12-200	1.31	5	2.07	5	1.64	5	1.18	5	1.06	5	0.45	6
4-8-13-200	5.88	7	11.11	7	9.31	8	6.54	8	6.00	9	9.70	62
4-8-14-400	55.62	7	75.70	7	39.81	8	27.77	8	15.59	9	19.89	62
7-6-11-100	1.00	6	2.27	6	2.42	6	2.29	6	1.22	7	5.38	54
7-6-12-500	1.80	5	3.08	5	3.62	5	3.23	5	1.80	5	1.64	8
7-6-13-500	4.56	5	8.85	5	7.34	5	5.86	5	4.48	6	11.96	30
7-6-14-1000	30.29	5	35.54	5	27.04	5	24.58	5	12.57	6	30.26	38

- . . . or (almost) uniformly worst (save for $\eta = 1$)
- but often strange things happen ($\eta = 1$ can even be best)

Easy Components

Decomposition of Fixed-Charge Multicommodity

- Multicommodity flow + arc design costs f_{ij} ($y_{ij} \in \{0,1\}$)
- $\mathcal{S}$ = extreme points of y ($2^{|A|}$ vertices of the unitary hypercube):

$$
\begin{array}{lll}
\min & \sum_{p\in\mathcal{P}} c_p f_p + \sum_{s\in\mathcal{S}} \left(\sum_{(i,j)\in A} f_{ij} \bar{y}^s_{ij} \right) \theta_s & \\
& \sum_{p\in\mathcal{P} \,:\, (i,j)\in p} f_p \le u_{ij} \sum_{s\in\mathcal{S}} \bar{y}^s_{ij} \theta_s & (i,j) \in A \\
& \sum_{p\in\mathcal{P}^k} f_p = d^k & k \in K \\
& f_p \ge 0 & p \in \mathcal{P} \\
& \sum_{s\in\mathcal{S}} \theta_s = 1 \quad , \quad \theta_s \ge 0 & s \in \mathcal{S}
\end{array}
$$

  Standard (weak) formulation used in Lagrangian approaches [15]

- Are you sure you're sane? Arguably not:

  replacing a $2n$ formulation with a 2^n one!

- . . . and with very long, dense rows

---

[15] Crainic, F., Gendron "Bundle-based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems" *Discrete Applied Mathematics* 2001

Fixed-Charge Multicommodity: even more disaggregated

- The unitary hypercube is a cartesian product: why not $\mathcal{S}^{ij} = \{0, 1\}$?
- $y_{ij} \longrightarrow 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1} \;,\; \theta^{ij,0} + \theta^{ij,1} = 1 \;,\; \theta^{ij,0} \geq 0 \;,\; \theta^{ij,1} \geq 0.$

$$\boxed{y_{ij} \in [\, 0 \,,\, 1 \,]}$$

Fixed-Charge Multicommodity: even more disaggregated

- The unitary hypercube is a cartesian product: why not $\mathcal{S}^{ij} = \{0, 1\}$?
- $y_{ij} \longrightarrow 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1}$, $\theta^{ij,0} + \theta^{ij,1} = 1$, $\theta^{ij,0} \geq 0$, $\theta^{ij,1} \geq 0$.

  $\boxed{y_{ij} \in [\,0\,,\,1\,]}$ (no, . . . really?!)

- Arc-path formulation with original arc design variables

$$
\begin{array}{lll}
\min & \sum_{p \in \mathcal{P}} c_p f_p + \sum_{(i,j) \in A} f_{ij} y_{ij} & \\
& \sum_{p \in \mathcal{P} \,:\, (i,j) \in p} f_p \leq u_{ij} y_{ij} & (i,j) \in A \\
& \sum_{p \in \mathcal{P}^k} f_p = d^k & k \in K \\
& f_p \geq 0 & p \in \mathcal{P} \\
& y_{ij} \in [0,1] & (i,j) \in A
\end{array}
$$

- Only generate the right variables

Is it always this easy?

- No: what if one had, say,

$$\sum_{(i,j)\in A} y_{ij} \leq r \qquad ?$$

- Design (y) subproblem can no longer be disaggregated
- But, one could write the arc-path formulation in that case, too
- And could add that constraint to the master problem
- Can this be stabilized? Of course it can [16]

---

[16] F., Gorgone "Bundle methods for sum-functions with "easy" components: applications to multicommodity network design" *Mathematical Programming?* 2013

Stabilized decomposition with "easy components"

- f Lagrangian function of structured optimization problem

$$(\Pi) \quad \max \left\{ c_1 x_1 + c_2(x_2) \ : \ x_1 \in X^1 \ , \ G(x_2) \leq g \ , \ A_1 x_1 + A_2 x_2 = b \right\}$$

i.e., $f(y) = f^1(y) + f^2(y)(-yb)$ where

$$f^1(\bar{y}) = \max \left\{ (c_1 - \bar{y} A_1) x_1 \ : \ x_1 \in X^1 \right\}$$

"easy for some reason" (efficient but "totally obscure" black box)

$$f^2(\bar{y}) = \max \left\{ c_2(x_2) - (\bar{y} A_2) x_2 \ : \ G(x_2) \leq g \right\}$$

"easy because a compact convex formulation is known"

Stabilized decomposition with "easy components"

- f Lagrangian function of structured optimization problem

  $(\Pi)\quad \max\left\{ c_1x_1 + c_2(x_2) \ : \ x_1 \in X^1 \ , \ G(x_2) \leq g \ , \ A_1x_1 + A_2x_2 = b \right\}$

  i.e., $f(y) = f^1(y) + f^2(y)(-yb)$ where

  $$f^1(\bar{y}) = \max\left\{ (c_1 - \bar{y}A_1)x_1 \ : \ x_1 \in X^1 \right\}$$

  "easy for some reason" (efficient but "totally obscure" black box)

  $$f^2(\bar{y}) = \max\left\{ c_2(x_2) - (\bar{y}A_2)x_2 \ : \ G(x_2) \leq g \right\}$$

  "easy because a compact convex formulation is known"

- Usual approach: disregard differences
  Better idea: treat "easy" components specially
- In practice: insert "full" description of f^2 in the master problem
- Master problem size may increase (at the beginning), but "perfect" information is known

"Easy components" in formulæ

- Dual master problem: abstract form

$$(\Delta_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \min \left\{ b(\bar{y}+d) + f^1_{\mathcal{B}}(\bar{y}+d) + f^2(\bar{x}+d) + \mathcal{D}(d) \right\}$$

- Primal master problem: abstract form

$$(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \max \begin{cases} c_1 x_1 + c_2(x_2) + \bar{y} z - \mathcal{D}^*(-z) \\ z = b - A_1 x_1 - A_2 x_2 \\ x_1 \in conv(\mathcal{B}) \quad , \quad x_2 \in X^2 \end{cases}$$

and implementable form

$$(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \max \begin{cases} c_1 \left(\sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta_{\bar{x}_1} \right) + c_2(x_2) + \bar{y} z - \mathcal{D}^*(-z) \\ z = b - A_1 \left(\sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta_{\bar{x}_1} \right) - A_2 x_2 \\ \sum_{\bar{x}_1 \in \mathcal{B}} \theta_{\bar{x}_1} = 1 \quad , \quad G(x_2) \leq g \end{cases} \tag{14}$$

- Barring some details (do not translate $f^1_{\mathcal{B}}$), everything works.

Extensions, details

- Multiple easy/hard components: trivial
- Constrained case: $y \in Y = \{\, y \;:\; Hy \leq h \,\}$

$$(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \max \begin{cases} c_1 \left(\sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta_{\bar{x}_1} \right) + c_2(x_2) + \omega h + \bar{y} z - \mathcal{D}^*(-z) \\ z = b + \omega H - A_1 \left(\sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta_{\bar{x}_1} \right) - A_2 x_2 \\ \sum_{\bar{x}_1 \in \mathcal{B}} \theta_{\bar{x}_1} = 1 \quad , \quad G(x_2) \leq g \quad , \quad \omega \geq 0 \end{cases}$$

- Global lower bound $l \leq \min f$: (c_2 and G linear)

$$(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \max \begin{cases} c_1 \sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta'_{\bar{x}_1} + c_2 x'_2 - l(1-\rho) + \bar{y} z - \mathcal{D}^*(-z) \\ z = \rho b - A_1 \sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta'_{\bar{x}_1} - A_2 x'_2 \\ \sum_{\bar{x}_1 \in \mathcal{B}} \theta'_{\bar{x}_1} = \rho \;\; , \;\; G x'_2 \leq \rho g \;\; , \;\; \theta' \geq 0 \;\; , \;\; \rho \in [0,1] \end{cases}$$

$(\theta = \theta'/\rho,\ x_2 = x'_2/\rho)$

Computational results

- Several possible options:
  - fully aggregated (FA)
  - partly disaggregated with easy y (PDE)
  - disaggregated with difficult y (DD)
  - disaggregated with easy y (DE)
- Stabilizing terms: $\|\cdot\|_\infty$, $\|\cdot\|_2^2$ only for (FA) (exploiting [17])
- With (strong) or without (weak) forcing constraints

$$0 \le c_{ij}^k \le u_{ij}^k y_{ij} \qquad (i,j) \in A\,,\ k \in K$$

(very many, so dynamic generation [18,19] needed)

- `http://www.di.unipi.it/optimize/Data/MMCF.html#Canad`

group	1	2	3	4	5	6	7	8	9	10	11	12
$\|N\|$	20	20	20	20	30	30	30	30	50	50	50	50
$\|A\|$	300	300	300	300	600	600	600	600	1200	1200	1200	1200
$\|K\|$	100	200	400	800	100	200	400	800	100	200	400	800

[17] F. "Solving semidefinite quadratic problems within nonsmooth optimization algorithms" *Computers & O.R.* 1996

[18] F., Lodi, Rinaldi "New approaches for optimizing over the semimetric polytope" *Mathematical Programming* 2005

[19] Belloni, Sagastizábal "Dynamic Bundle Methods" *Mathematical Programming* 2009

Computational results – weak formulation

DE			PDE				DD				FA-1				FA-2			
time	f	iter	time	f	iter	gap	time	f	iter	gap	time	f	iter	gap	time	f	iter	gap
0.04	0.00	5	0.03	0.01	6		557	2.54	6200	1e-7	979	3.97	9105	1e-3	7.64	0.75	2383	1e-7
0.08	0.01	6	0.08	0.01	12		772	2.94	3153	6e-3	1000	4.43	4772	3e-2	14.24	1.37	1931	6e-9
0.25	0.01	7	0.57	0.12	52	1e-7	739	2.79	1365	2e-7	862	10.57	5579	3e-3	12.66	1.99	1117	5e-7
0.64	0.03	7	1.06	0.23	50	3e-7	1000	2.27	482	9e-3	1000	14.49	3201	8e-3	42.38	7.74	1714	7e-7
0.10	0.01	7	0.30	0.03	39		665	4.92	5799	4e-3	945	6.15	7538	8e-3	4.12	0.50	834	3e-7
0.25	0.02	10	1.81	0.21	122		498	3.37	1899	7e-8	808	9.76	5599	3e-3	6.36	1.06	664	1e-6
0.45	0.04	8	20.56	1.93	483	2e-7	1000	1.81	415	2e-2	1000	2.58	638	5e-2	134.49	15.00	3795	6e-7
1.10	0.08	9	5.17	1.09	120	1e-7	1000	3.48	378	2e-2	1000	10.08	1134	4e-2	126.29	26.19	2905	8e-7
0.34	0.02	11	34.80	0.78	449	5e-9	1000	1.39	746	5e-3	1000	2.23	1205	4e-2	28.92	2.77	1630	1e-6
0.42	0.05	9	2.39	0.26	89		1000	6.23	1647	3e-2	1000	8.51	2343	5e-2	32.77	5.26	1414	8e-7
0.99	0.10	11	16.03	2.34	271	1e-7	1000	6.18	717	2e-2	1000	11.31	1321	4e-2	80.05	16.48	1848	8e-7
2.19	0.18	10	124.38	13.95	811	6e-7	1000	5.05	278	2e-2	1000	14.63	838	6e-2	233.40	50.47	2851	8e-7

- relative accuracy = 1e−6, maximum running time = 1000 seconds
- all things being equal, $\|\cdot\|_2^2$ trounces $\|\cdot\|_\infty$
- PDE better than DD, DE demolishes everything else
- Master problem time largely preponderant (no trick like [20])

[20] Cappanera, F. "Symmetric and asymmetric parallelization of a cost-decomposition algorithm for multi-commodity flow problems" *INFORMS Journal on Computing* 2003

Computational results – weak formulation – accuracy

Cplex						DE		FA-2	
primal	dual	barrier	p.net.	d.net.	auto	1e-6	1e-12	1e-6	1e-12
0.30	0.13	8.73	0.18	0.23	0.36	0.04	0.04	7.64	7.74
0.89	0.90	21.25	0.58	1.95	2.40	0.08	0.08	14.24	14.37
3.04	10.22	76.24	2.24	16.32	25.44	0.25	0.26	12.66	13.13
8.21	16.56	151.14	4.62	27.58	44.79	0.64	0.64	42.38	49.18
1.09	4.98	42.57	0.74	6.88	10.62	0.10	0.10	4.12	4.19
3.28	24.68	135.57	2.77	29.46	69.86	0.25	0.26	6.36	7.94
53.25	22.58	417.10	8.96	51.45	55.86	0.45	0.45	134.49	137.41
18.74	67.24	1115.22	10.56	99.96	177.40	1.10	1.10	126.29	163.88
19.98	84.33	303.29	3.92	112.71	187.37	0.34	0.35	28.92	42.71
7.89	82.64	583.52	18.60	259.65	309.74	0.42	0.42	32.77	40.60
38.09	230.79	1952.75	15.85	325.33	690.30	0.99	0.99	80.05	108.94
586.07	459.49	3586.63	51.71	738.23	1266.87	2.19	2.19	233.40	1789.08

- Cplex accuracy is 1e-12, except for barrier where is 1e-10
- Decomposition approaches tested with both 1e-6 and 1e-12
- DE trounces the best of Cplex (primal network) by an order of magnitude, all the rest by much more, at the same accuracy

Computational results – strong formulation – tuning

Cplex		DE	
static	dynamic	static	dynamic
54	10	44	32
315	54	233	48
1539	112	1234	29
2789	458	2227	65

Comparison of static and dynamic constraint handling

DE			PDE			DD			FA-1			FA-2		
time	iter	gap	time	iter	gap	time	iter	gap	time	iter	gap	time	iter	gap
32	77	1e-7	1000	2980	2e-2	1000	2714	2e-1	1000	1990	2e-1	410	14880	9e-7
48	30	3e-7	3000	2896	6e-2	3000	3720	7e-2	3000	7351	2e-1	1855	11141	3e-6
29	24	2e-7	9000	8370	2e-2	9000	5061	5e-2	9000	10918	1e-1	1254	9035	2e-6
65	20	3e-8	27000	5618	3e-2	27000	2148	4e-2	27000	5293	8e-2	1732	12940	1e-6

(Partial) results for the strong formulation

- Dynamic (lazy) constraints handling is necessary (even for `Cplex`)
- Even allowing long running time, PDE, DD, and FA-1 have the chance of a snowball in hell

Computational results – strong formulation – more tuning

- To be efficient, you have to let information accumulate!
- Optimal setting: maximum $|\mathcal{B}| = 50 \cdot |K|$, constraints violation checked at every iteration, constraints never removed
- Experiments: $|\mathcal{B}| = 20 \cdot |K|$, constraints checked every 10 iterations and removed if $x_i = 0$ for 20 consecutive iterations

opt			$20 \cdot \lvert K \rvert$			Rmv $= 20$			Sep $= 10$		
time	it	gap	time	it	gap	time	it	gap	time	it	gap
31.69	77	1e-7	289.41	841	7e-7	104.60	218	2e-7	72.96	194	1e-6
47.53	30	3e-7	3000.76	1585	3e-4	1564.82	803	4e-5	363.67	159	3e-7
28.98	24	2e-7	1125.93	726	4e-7	2585.05	796	1e-6	141.61	65	1e-6
65.31	20	3e-8	81.33	20	3e-8	17415.68	2121	8e-5	669.34	78	5e-7

- No, no, no!
- The "combinatorial tail" have to start soon, it is easily destroyed

Computational results – strong formulation – accuracy

1e-6					1e-8			1e-10			1e-12			
time	f	add	iter	gap	time	iter	gap	time	iter	gap	time	f	add	iter
31.69	0.05	0.96	77	1e-7	57.73	143	4e-9	62.07	170	3e-11	63.78	0.11	1.10	181
47.53	0.04	2.04	30	3e-7	51.22	33	2e-9	51.37	33		51.38	0.05	2.06	33
28.98	0.07	2.70	24	2e-7	29.15	25		29.15	25		29.16	0.07	2.74	25
65.31	0.14	6.58	20	3e-8	65.67	21		65.68	21		65.69	0.15	6.61	21
25.93	0.04	0.89	47	8e-8	28.28	51	3e-9	32.00	57		32.00	0.06	0.93	57
27.97	0.09	1.48	36	4e-7	55.43	51	4e-10	56.01	52	1e-11	56.28	0.12	1.60	52
20.80	0.09	1.80	21	2e-8	20.84	21	2e-9	25.69	24		25.69	0.11	1.84	24
132.60	0.24	10.03	23	8e-8	132.74	23		132.76	23		132.78	0.24	10.09	23
2.47	0.06	0.48	26	2e-10	2.47	26	2e-10	2.57	27	3e-12	2.66	0.06	0.49	27
245.91	0.26	4.18	59	1e-7	295.56	72	4e-9	333.22	84	2e-11	337.38	0.39	4.54	86
283.71	0.43	7.24	39	7e-8	442.56	55	2e-9	506.83	63	5e-12	507.52	0.71	7.78	63
241.84	0.52	11.85	24	2e-11	241.88	24	2e-11	241.92	24	2e-11	253.59	0.55	11.98	25

- Four accuracy settings: 1e-6, 1e-8, 1e-10, 1e-12
- Not quite as spectacular as in the weak case, but double precision in $\leq$ double time
- "add" = time for checking constraints $\gg$ time for f computation!

Computational results – strong formulation

Cplex				DE		FA–2					FA–V				
primal	dual	net.	barr.	1e-6	1e-12	time	f	add	it	gap	time	f	add	it	gap
12	10	11	15	32	64	410	12	7	14880	9e-7	3	0.6	0.5	875	9e-3
64	53	61	71	48	51	1855	19	16	11141	3e-6	6	1.2	1.2	842	2e-2
139	114	132	157	29	29	1254	32	20	9035	1e-6	12	2.3	2.2	796	3e-2
559	456	531	587	65	66	1732	100	67	12940	1e-6	26	5.1	5.0	760	4e-2
46	39	43	60	26	32	322	12	10	10320	1e-6	6	0.9	1.1	871	8e-3
147	132	144	209	28	56	294	15	9	5300	1e-6	12	2.1	2.4	831	9e-3
509	301	478	648	21	26	5033	169	155	27231	1e-6	26	4.5	5.4	794	3e-3
2329	1930	2302	2590	133	133	3122	192	169	14547	1e-6	51	8.6	10.6	760	4e-2
196	131	156	304	2	3	344	20	12	7169	1e-6	12	2.0	2.3	827	3e-3
926	708	862	1174	246	337	2256	111	118	17034	2e-5	29	5.0	6.1	869	1e-2
2706	2167	2542	3272	284	508	5475	192	249	15061	3e-6	58	9.2	13.0	817	2e-2
11156	8908	11675	11683	242	253	11863	349	413	13953	1e-6	109	16.7	24.1	765	2e-2

- Fa–V: a FA with volume algorithm, quick but too coarse
- Sep $= 100$ for FA–2, Sep $= 10$ for FA–V: computing x^* by convex combination much faster, bottleneck for DD
- More than an order of magnitude to Cplex as $|A|$ and/or $|K|$ grows

Last comment: similar ideas

- Nonlinear multicommodity routing:

$$\min \left\{ \sum_{(i,j)\in A} \frac{y_{ij}}{1-y_{ij}} \ : \ (3)-(6)\ ,\ y \in [0,1]^{|A|} \right\}$$

  with classical (convex) Kleinrock delay function

- Decomposes into $|K|$ flows $+ |A|$ simple convex subproblems

- Specialized models of $|A|$ convex functions using the conjugate

- Specialized treatment of these "easy" C^2 functions with Newton model instead of the cutting-plane model [21]

- Substantially improved performances

[21] Lemaréchal, Ororou, Petrou "A bundle-type algorithm for routing in telecommunication data networks" *Computational Optimization and Applications* 2009

Structured Decomposition

Structured Decomposition

- Came out for a different (still multicommodity) problem [22]
- D-W $\equiv$ reformulation of X always in the same form ...

Structured Decomposition

- Came out for a different (still multicommodity) problem [22]
- D-W $\equiv$ reformulation of X always in the same form ...
  or not, as we have already seen. But we can do better:
  - Assumption 1: alternative Formulation of "easy" set

$$X = \left\{ x = C\theta \ : \ \Gamma\theta \leq \gamma \right\}$$

  - Assumption 2: $\mathcal{B}$ subset of rows and columns, padding with zeroes

$$\Gamma_{\mathcal{B}}\bar{\theta}_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \implies \Gamma\left[\bar{\theta}_{\mathcal{B}} \ , \ 0 \right] \leq \gamma$$

$$\implies X_{\mathcal{B}} = \left\{ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ : \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \subseteq X$$

  - Assumption 3: easy update of rows and columns

    Given $\mathcal{B}$, $\bar{x} \in X$, $\bar{x} \notin X_{\mathcal{B}}$, it is "easy" to find $\mathcal{B}' \supset \mathcal{B}$

    $(\implies \Gamma_{\mathcal{B}'}, \gamma_{\mathcal{B}'})$ such that $\exists\ \mathcal{B}'' \supseteq \mathcal{B}'$ such that $\bar{x} \in X_{\mathcal{B}''}$.

[22] F., Gendron "0-1 reformulations of the multicommodity capacitated network design problem" *Discrete Applied Math.* 2009

The Structured Dantzig-Wolfe Algorithm

- Structured master problem $\equiv$ structured model

$$(\Pi_{\mathcal{B}}) \qquad \max\left\{ cx \ : \ Ax = b \ , \ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\}$$

$$f_{\mathcal{B}}(y) \ = \ \max\{ \ (c - yA)x + xb \ : \ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \ \}$$

$\langle$ initialize $\mathcal{B}$ $\rangle$;
`repeat`
  $\langle$ solve $(\Pi_{\mathcal{B}})$ for x^*, y^* (duals of $Ax = b$); $v^* = cx^*$ $\rangle$;
  $\bar{x} = \text{argmin}\ \{ \ (c - y^*A)x \ : \ x \in X \ \}$;
  $\langle$ update $\mathcal{B}$ as in Assumption 3 $\rangle$;
`until` $v^* < c\bar{x} + y^*(b - A\bar{x})$

- Finitely terminates with an optimal solution, even if (proper) removal from $\mathcal{B}$ is allowed, X is non compact and $\mathcal{B} = \emptyset$ at start (Phase 0)
- The subproblem to be solved is identical to that of DW
- Requires ($\Longrightarrow$ exploits) extra information on the structure
- Master problem with any structure, possibly much larger

Stabilizing the Structured Dantzig-Wolfe Algorithm

- Exactly the same as stabilizing DW: stabilized master problem

$$(\Delta_{\mathcal{B},\bar{y},\mathcal{D}}) \qquad \min \left\{ f_{\mathcal{B}}(\bar{y} + d) + \mathcal{D}(d) \right\}$$

  except $f_{\mathcal{B}}$ is a different model of f (not the cutting plane one)

- Even simpler from the primal viewpoint [23]:

$$\max \left\{ cx + \bar{y}z - \mathcal{D}^*(z) \,:\, z = Ax - b \,,\; x = C_{\mathcal{B}}\theta_{\mathcal{B}} \,,\, \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\}$$

- With proper choice of $\mathcal{D}$, still a(sparsely structured) Linear Program
- Dual optimal variables of "$z = Ax - b$" still give d^*, ...
- How to move $\bar{y}$, handle t, handle $\mathcal{B}$: basically as in [10], actually even somewhat simpler because $\mathcal{B}$ is inherently finite
- Funnily, aggregation $\mathcal{B} = \mathcal{B} \cup \{ x^* \}$ is also possible, up to

$$\mathcal{B} = \{ x^* \} \equiv \text{"poorman" method}$$

  although clearly contrary to the spirit of S^{2}DW

---

[23] F., Gendron "A Stabilized Structured Dantzig-Wolfe Decomposition Method" *Mathematical Programming* 2013

Structured Decomposition for Multicommodity Flows

- All nice and well, but how can we come up with a $x = C\theta$?
- Surprisingly simple: use the node-arc formulation
- Start with "empty graph", find paths: if a node/arc is missing, add it
- Intermediate formulation between node-arc and arc-path
- Would seem to generalize to many other network-structured problems
- Current implementation heavily relies on Cplex preprocessor
  it may be preferable to do the path splitting by hand
- Current implementation is not stabilized at all

(Preliminary) Computational results

- Ad-hoc code (including in general Bundle nontrivial, but possible)
- No stabilization (but probably none needed)
- Still using Cplex as main driving force
- Comparing also against direct use of Cplex (tuned)
- Exactly the same subproblem solver (`FiOracle`)
- Surely can be improved a lot (e.g. explicit graph operations)
- Same instances, same machine

Computational Results: Planar & Grid Instances

	0		*		SDW		Cplex
	time	it.	time	it.	time	it.	time
grid7	2.5	12	2.12	14	1.29	9	54.73
grid8	18.52	18	18.33	19	23.81	12	1745.65
grid9	36.04	15	36.04	15	193.53	12	***
grid10	54.51	15	54.51	15	596.83	13	***
grid12	61.64	11	61.24	10	881.37	11	***
grid14	433.64	11	230.66	11	6086.84	11	***
planar100	2.16	14	1.42	13	2.66	8	43.90
planar150	25.75	17	25.75	17	183.94	11	4239.98
planar300	21.34	13	21.34	13	112.87	9	5127.74
planar500	15.27	11	10.84	11	25.16	7	***

- *** = out of time limit (6400 seconds): Cplex clearly worst
- SDW seldom competitive here, although much better than Cplex
- $\eta = 0$ not a bad choice overall, but not necessarily best

Computational Results: Goto & Mnetgen Instances

	0 = *		SDW		Cplex
	time	it.	time	it.	time
Goto6-100	1.05	25	0.60	11	0.67
Goto6-400	1.45	15	2.42	14	14.22
Goto6-800	2.41	12	5.54	15	64.09
Goto8-10	2.96	75	0.11	8	0.11
Goto8-100	3.43	21	1.45	14	5.63
Goto8-400	5.88	16	11.12	17	105.13
Goto8-800	3.12	11	17.23	18	326.01
128-32	17.66	57	3.90	6	0.32
128-32	57.23	46	15.08	6	0.87
128-64	95.45	34	32.66	7	1.61
128-128	5.68	109	0.25	5	0.05
256-8	31.65	140	0.80	6	0.07
256-16	146.37	148	4.97	6	0.28
256-32	400.59	117	23.95	6	1.07
256-64	563.66	86	61.45	7	1.69

- SDW is not often the best, but it is never the worst

Computational Results: Waxman [& Rmnet] Instances

	0		0.8 = *		SDW		Cplex
	time	it.	time	it.	time	it.	time
W-50	1.43	3	0.03	3	0.32	7	1.12
W-100-6	1.53	2	0.04	2	0.39	7	1.20
W-100-10	1.34	3	0.22	3	1.11	6	3.14
W-100	1.50	2	0.72	2	0.86	2	22.49
W-150-6	2.44	2	0.63	2	2.93	6	33.82
W-150-10	1.23	3	0.14	2	3.54	4	10.38
W-150	3.23	3	0.67	3	2.14	3	52.21

- Er . . . Rmnet not ready yet, sorry (preliminary I said)
- When few paths (= iterations) are required, SDW can't help much
- Still better than using Cplex directly, though
- Often better than standard decomposition with non-optimal η
- Results on other applications much more promising [22,23,24]

[24] F., Finardi, Scuzziato "Decomposition Approaches to Large-Scale Stochastic Unit-Commitment Problems" *working paper*

Conclusions
(for good, this time)

Conclusions and (a lot of) future work

- Decomposition methods (DW, Benders') old ideas, well-understood
- Yet, by-the-book decomposition is often not effective enough
- Many possible ideas to improve on the standard approach
- Substantial issue: what works is "large" master problems so that "combinatorial tail" kicks in very quickly $\Longrightarrow$
  - Large master problem time
  - "Unstructured" master problems $\Longrightarrow$ general-purpose solvers
  - "Complicated" $\Longrightarrow$ costly stabilizing functions ($\|\cdot\|_2^2$)
  - Need to find modern equivalent of [17] to exploit the structure of an unstructured problem (perhaps less contradictory than is sounds [25])
- Other important idea: inexact solution of the subproblems [12]
- Huge challenge: make these techniques mainstream
- A possible way: automatic reformulation tools [26]

---

[25] Kiwiel "An alternating linearization bundle method for ... and nonlinear multicommodity flow problems" *Math. Prog.* 2013

[26] F., Perez Sanchez "Transforming mathematical models using declarative reformulation rules" *LNCS* 6683, 2011

Visit Pisa in September!

Come to AIRO 2015