Decomposition in Multicommodity Flows: Old Ideas and New Developments

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

NetOpt2013, Estoril

January 14th, 2013

Outline

A reasonably generic Multicommodity flow model

- Graph $G = (N, A)$, a reasonably generic Multicommodity flow model

$$\min \sum_{k \in K} \sum_{(i,j) \in A} d^k c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{1}$$

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

$$\sum_{k \in K} d^k x_{ij}^k \leq u_{ij} y_{ij} \qquad (i,j) \in A \tag{3}$$

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

$$y_{ij} \in \mathbb{N} \,/\, \{0,1\} \,/\, \mathbb{R} \qquad (i,j) \in A \tag{5}$$

- $K \equiv$ commodities $\equiv (s^k\ ,\ t^k\ ,\ d^k)$ (not completely generic)
- Countless many relevant special cases:
  - $f \leq 0 \Longrightarrow$ splittable/nonsplittable multicommodity routing
  - $y_{ij} \in \{0,1\} \Longrightarrow$ almost all graph design problems
  - $\sum_{k \in K} d^k \leq u_{ij}$ + bipartite graph $\Longrightarrow$ uncapacitated facility location
  - multiple node/arc capacities by graph transformations
- Countless many generalizations (extra constraints, nonlinearities, ...)

Multicommodity flow applications

- Pervasive structure in most of combinatorial optimization
- Interesting links with many hard problems (e.g. Max-Cut)
- Very many practical applications: logistic, transportation, telecommunications, energy, ...
- Extremely hard to solve in general: many difficult problems in one
- Very different cases:
  - transportation: very large (often time-space $\Longrightarrow$ acyclic) networks, "few" commodities
  - telecommunications: "small" (undirected) networks, very many ($O(|N|^2)$) commodities
- Even continuous versions "hard": very-large-scale LPs
- Many sources of structure $\Longrightarrow$ the paradise of decomposition [1,2]

---

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

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

(Very) Classical decomposition approaches

- Lagrangian relaxation [3] of linking constraints:
  - (3): $\Longrightarrow$ flow (shortest path) relaxation
  - (2): $\Longrightarrow$ knapsack relaxation
  - others possible
- Benders' decomposition [4] of linking variables:
  - design (y) variables are "naturally" linking
  - 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") [5]
- We will talk of Lagrange but some things can be extended to Benders

---

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

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

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

Dantzig-Wolfe decomposition

A bird's view of "structure"

- The general structure of our models:

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

where we know something about the combinatorial set X:

  - for sure we know some formulation, e.g., $X = \{ x \in \mathbb{Z}^n \ : \ Ex \leq d \}$
  - linearity used for simplicity, has some (but not paramount) impact, convexity is the issue
  - integrality a common (but not the only) nonconvex component
  - almost always $X = \bigotimes_{h \in \mathcal{K}} X^h$, i.e., $Ax = b$ are linking constraints (note that not always $K = \mathcal{K}$)

A bird's view of "structure"

- The general structure of our models:

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

  where we know something about the combinatorial set X:
  - for sure we know some formulation, e.g., $X = \{ x \in \mathbb{Z}^n \ : \ Ex \leq d \}$
  - linearity used for simplicity, has some (but not paramount) impact, convexity is the issue
  - integrality a common (but not the only) nonconvex component
  - almost always $X = \bigotimes_{h \in \mathcal{K}} X^h$, i.e., $Ax = b$ are linking constraints (note that not always $K = \mathcal{K}$)

- What do we know? Perhaps the least possible requirement is we know how to (relatively) efficiently optimize upon X
  - clearly, $X = \bigotimes_{h \in \mathcal{K}} X^h$ helps a lot here
  - special case: we know the best possible convex formulation

$$\tilde{E}x \leq \tilde{d} \quad \text{such that} \quad \{ x \in \mathbb{R}^n \ : \ \tilde{E}x \leq \tilde{d} \} = conv(X)$$

    except, practically all good formulations are too large
  - sometimes $\tilde{E}x \leq \tilde{d}$ can be generated piecemeal, but not always

What does this gives?

- The best possible (convex = solvable) relaxation

$$(\bar{\Pi}) \qquad \max \{ cx \ : \ Ax = b \ , \ x \in conv(X) \} \tag{6}$$

trivial if "by faces" representation $\tilde{E}$, $\tilde{d}$ known, workable, otherwise?

What does this gives?

- The best possible (convex = solvable) relaxation

$$(\bar{\Pi}) \qquad \max \{ \, cx \; : \; Ax = b \, , \; x \in conv(X) \, \} \qquad (6)$$

trivial if "by faces" representation $\tilde{E}$, $\tilde{d}$ known, workable, otherwise?

- Temporarily assume X compact: represent $conv(X)$ by points instead

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

then reformulate $(\bar{\Pi})$ in terms of the convex multipliers θ

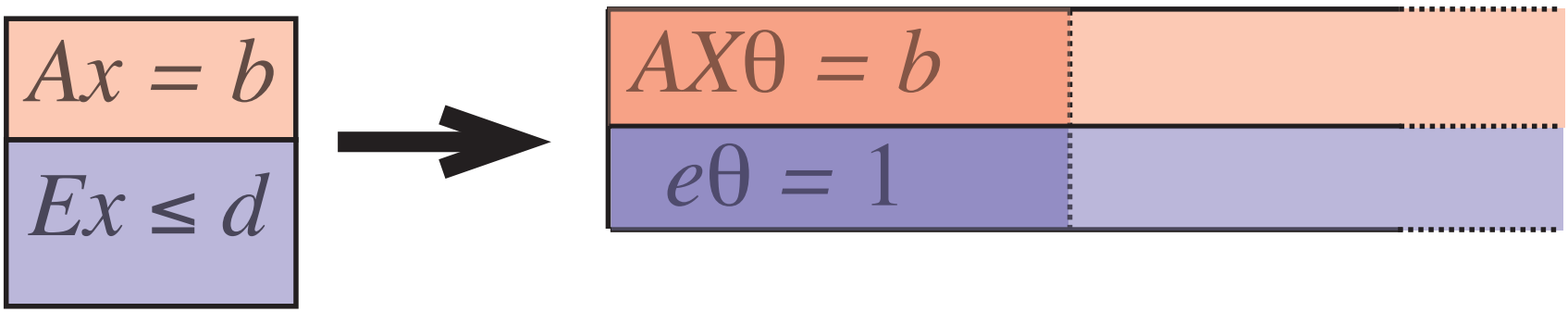
$$(\tilde{\Pi}) \qquad \left\{ \begin{array}{lll} \max & c \left(\sum_{\bar{x} \in X} \bar{x} \theta_{\bar{x}} \right) & \\ & A \left(\sum_{\bar{x} \in X} \bar{x} \theta_{\bar{x}} \right) = b & \\ & \sum_{\bar{x} \in X} \theta_{\bar{x}} = 1 \quad , \quad \theta_{\bar{x}} \geq 0 & \bar{x} \in X \end{array}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

- 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 ≡ Lagrangian relaxation

- Actually, only the vertices $V \subseteq X$ of $conv(X)$ are required
- Except, most often the number of vertices is too large



- But, if we can efficiently optimize over X, we can generate vertices
- $\mathcal{B} \subset X$ (small), solve restriction of $(\tilde{\Pi})$ with $X \to \mathcal{B}$, i.e.,

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

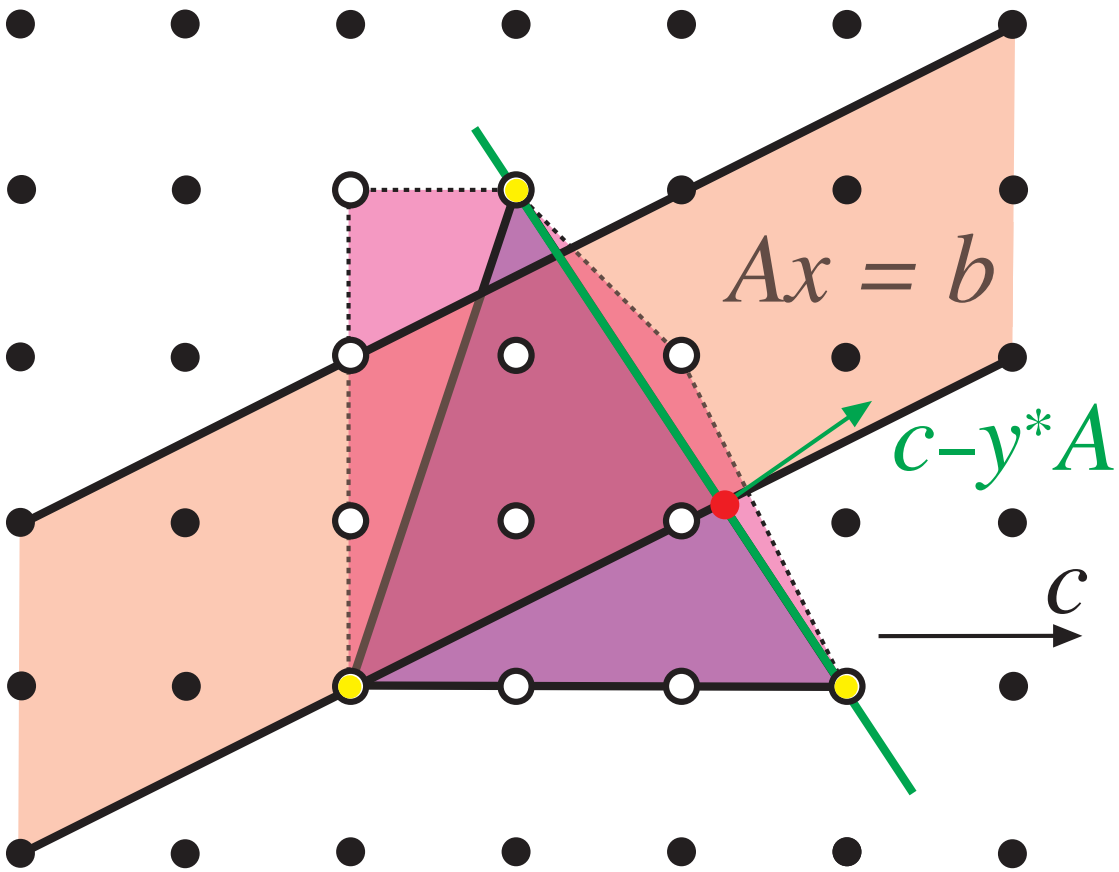
feed (partial) dual optimal solution y^* (of $Ax = b$) to pricing problem

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

a.k.a. Lagrangian relaxation

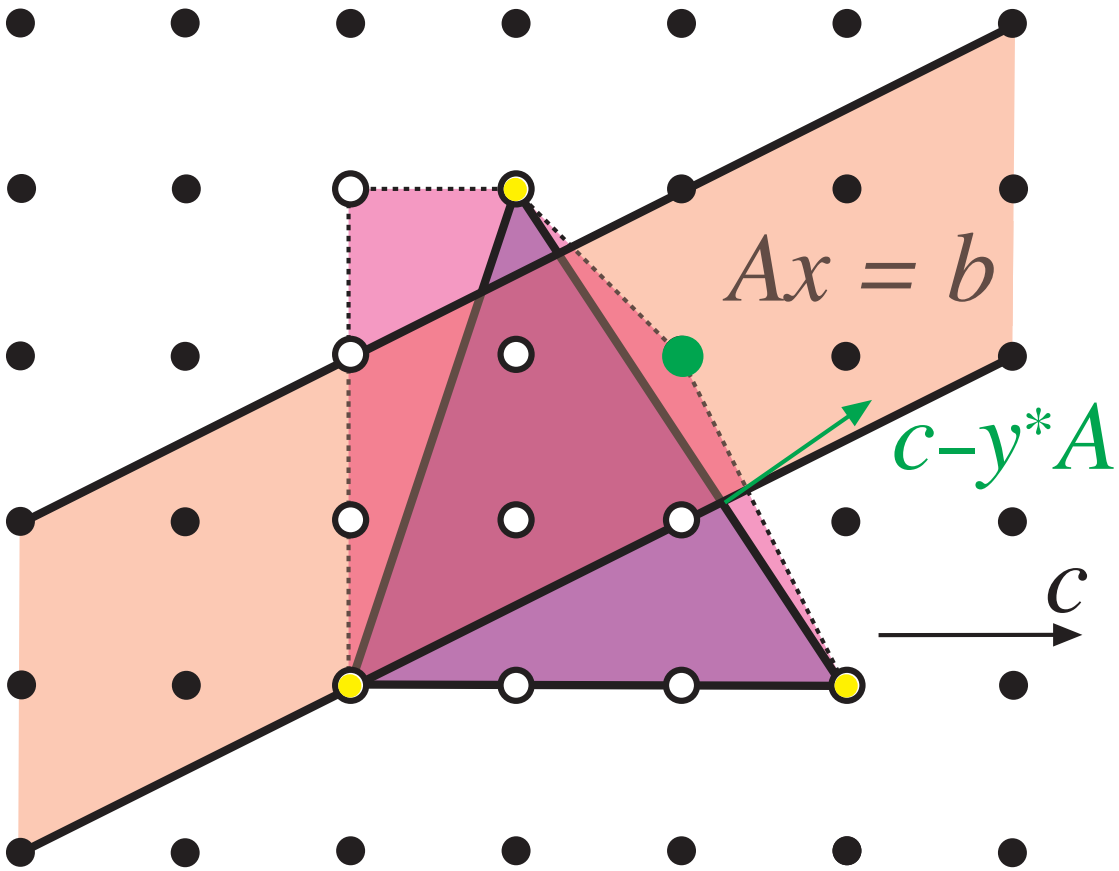
- Use primal optimal solution $\bar{x}$ of (Π_{y^*}) to enlarge $\mathcal{B}$

Geometry of Dantzig-Wolfe decomposition



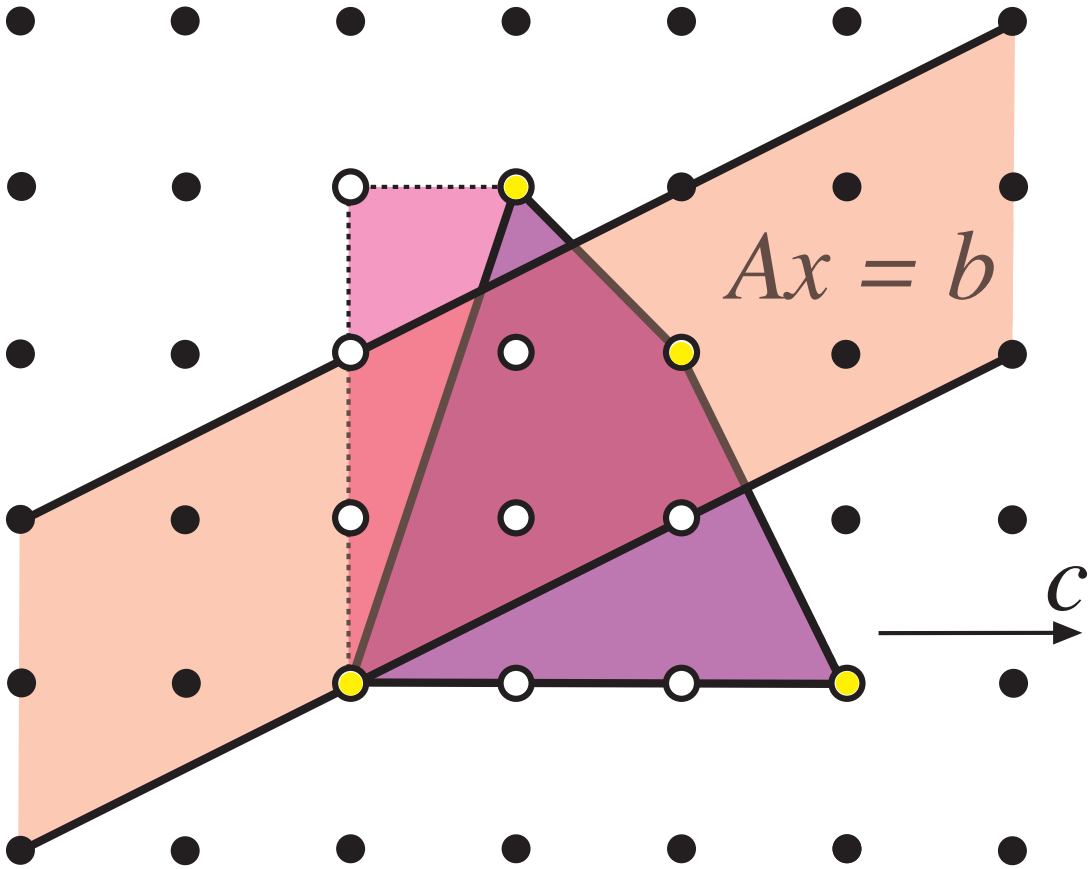
- $c - y^*A$ separates $conv(\mathcal{B}) \cap Ax = b$ from all $x \in X$ better than x^*

Geometry of Dantzig-Wolfe decomposition



- $c - y^*A$ 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 decomposition



- $c - y^*A$ 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

$$(\Delta) \qquad \min \left\{ f(y) = \max \left\{ cx + y(b - Ax) \ : \ x \in X \right\} \right\} \qquad (7)$$

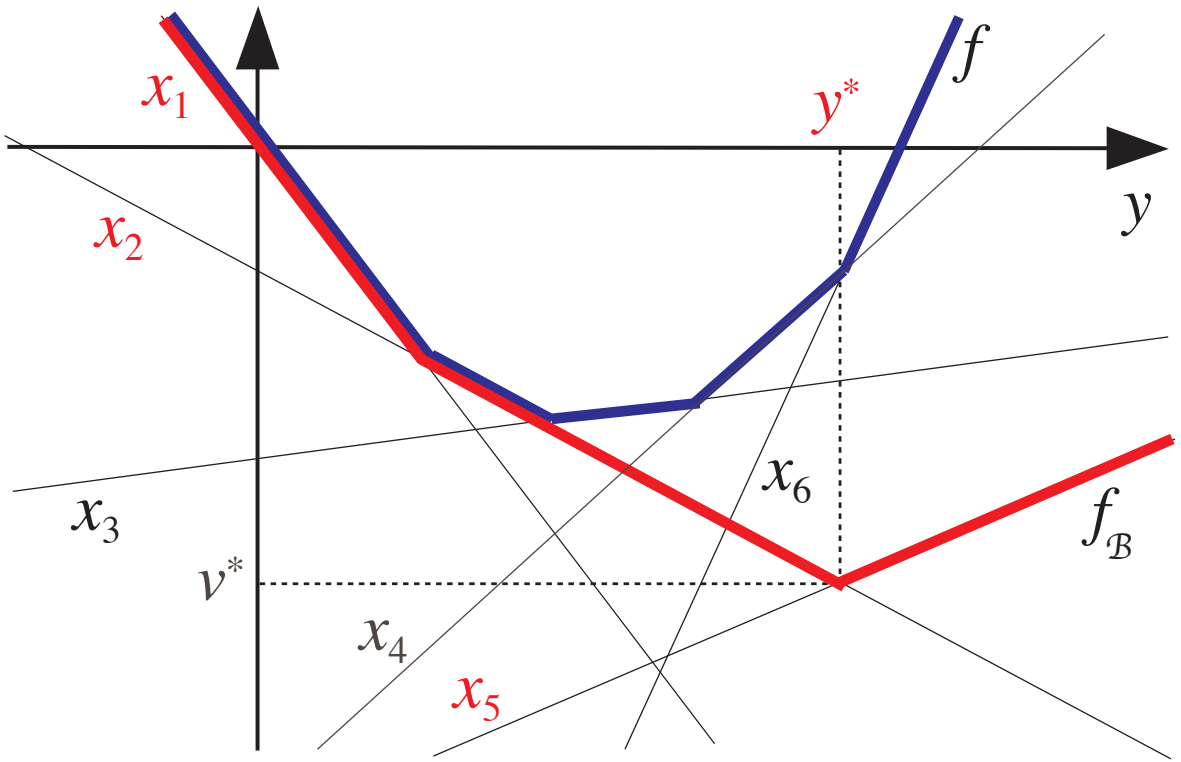
  that is equivalent to $(\bar{\Pi})$, $(\tilde{\Pi})$

- Dantzig-Wolfe decomposition $\equiv$ Cutting Plane approach to (Δ) [6]

---

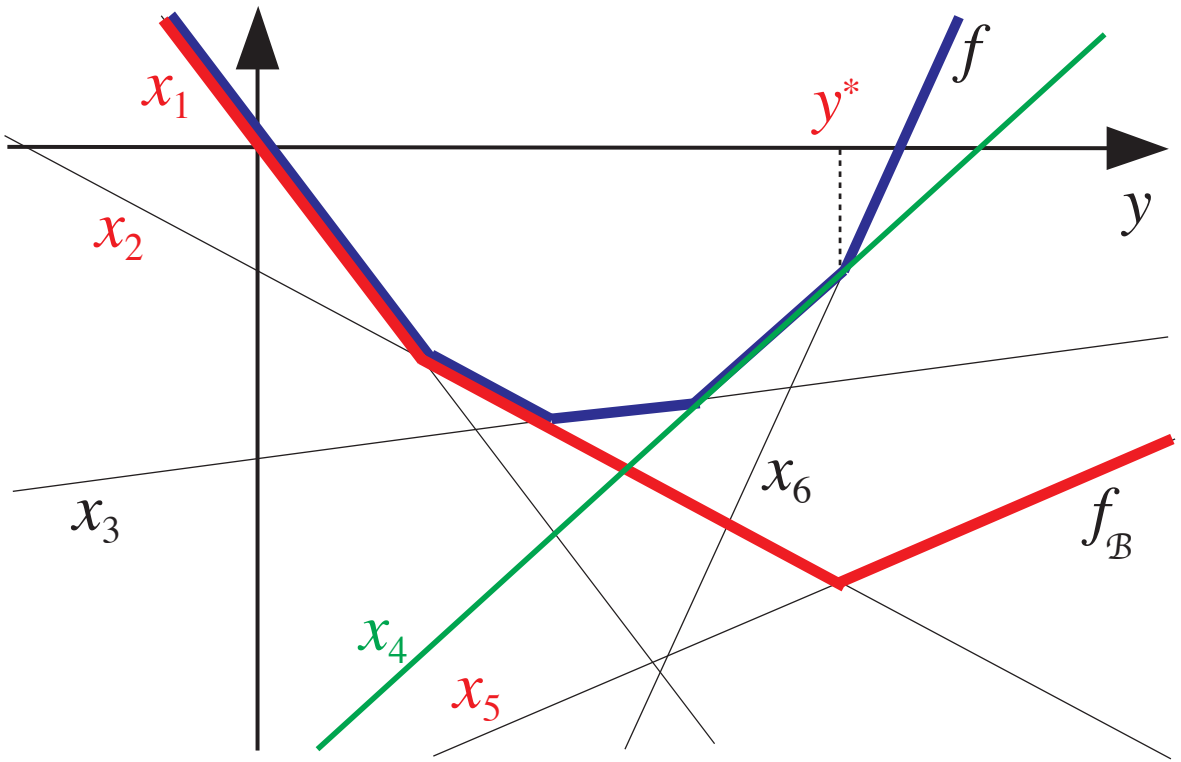
[6] Kelley "The Cutting-Plane Method for Solving Convex Programs" *Journal of the SIAM* 1960

Geometry of the Lagrangian dual



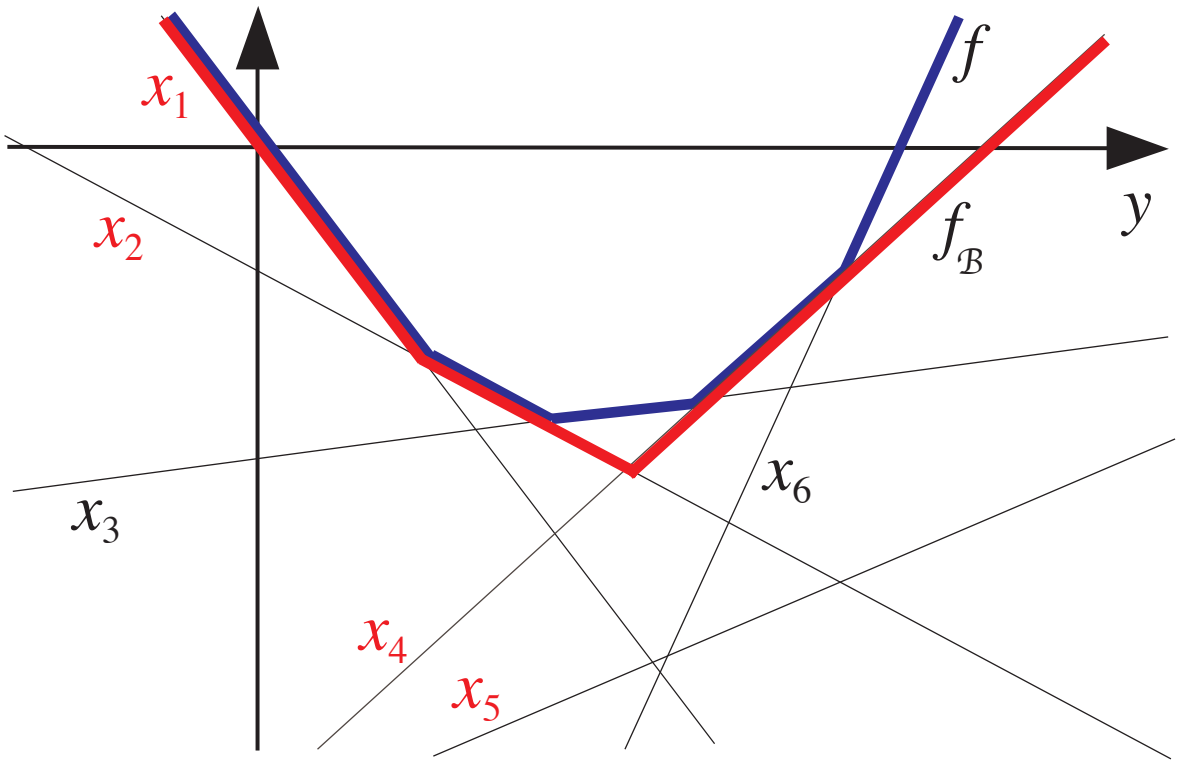
- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$

Geometry of the Lagrangian dual



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

Geometry of the Lagrangian dual

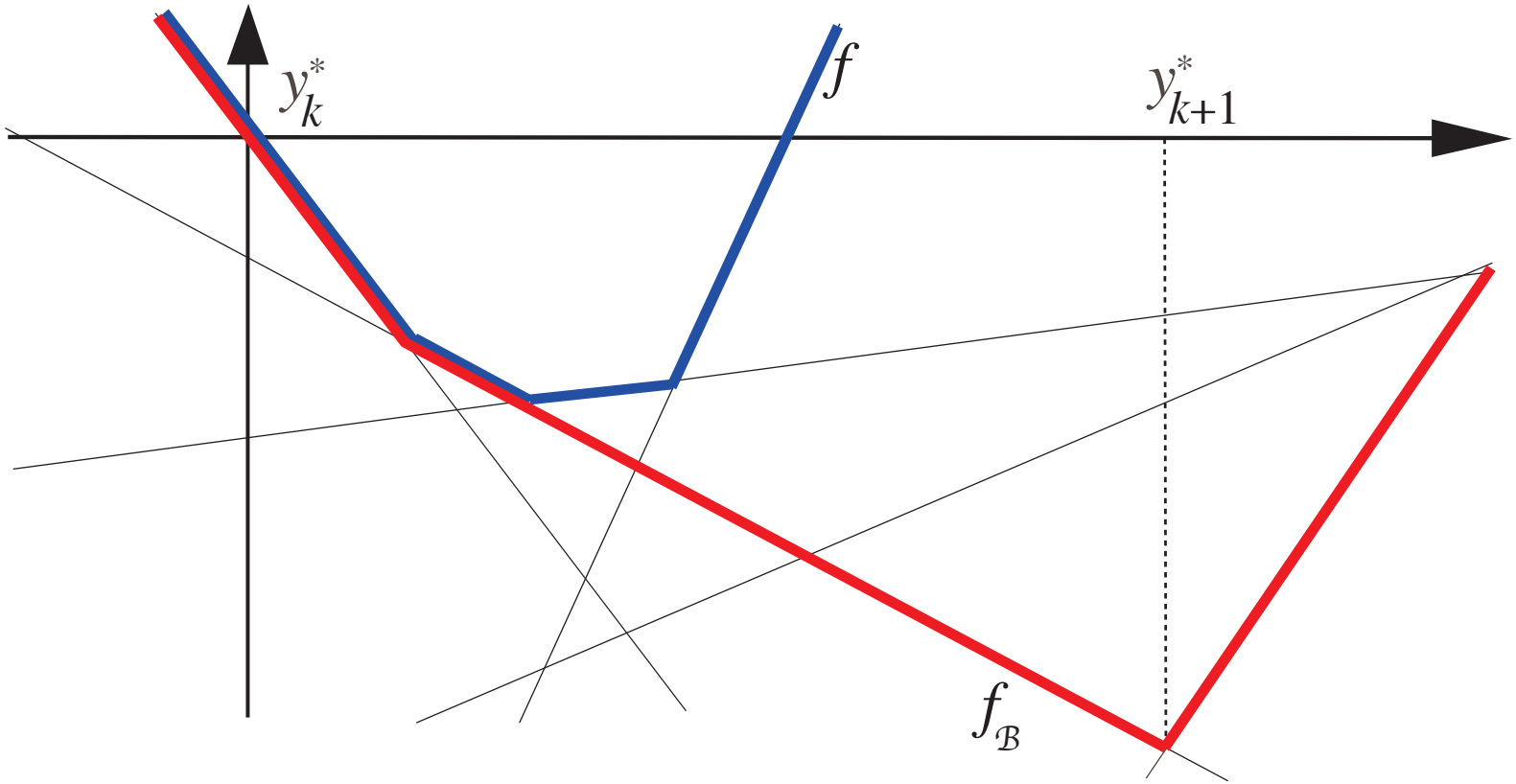


- $v^* = f_{\mathcal{B}}(y^*)$ lower bound on $v(\Pi_{\mathcal{B}})$
- Optimal solution $\bar{x}$ gives separator between (v^*, y^*) and *epi* f
- $(c\bar{x}, A\bar{x})$ = new row in $(\Delta_{\mathcal{B}})$ (subgradient of f at y^*)

Stabilization

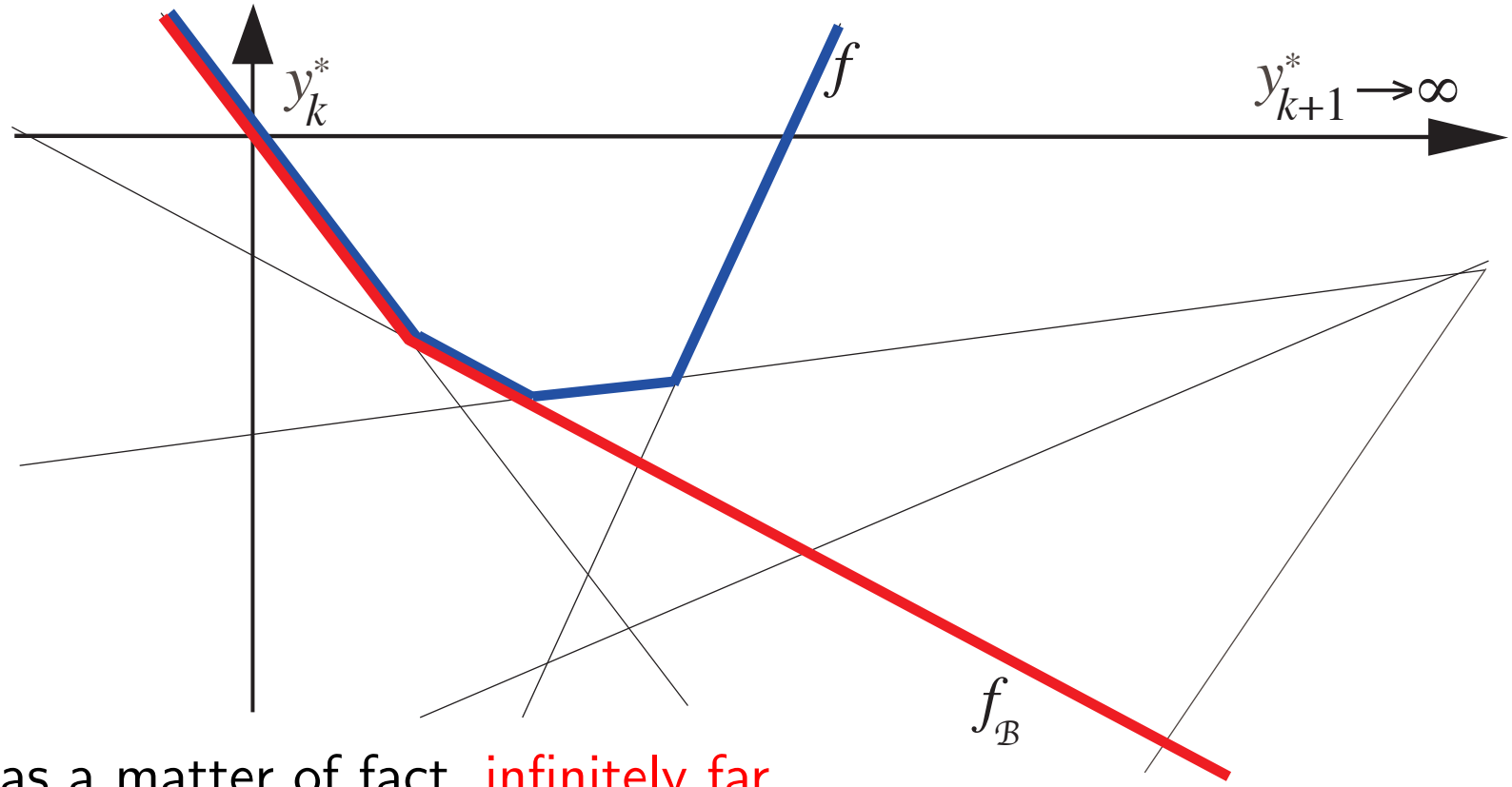
Issue with the approach: instability

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



Issue with the approach: instability

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



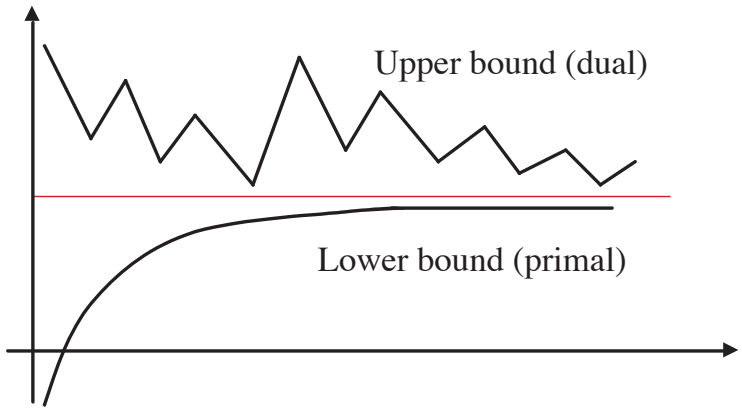
. . . as a matter of fact, infinitely far

- $(\Pi_{\mathcal{B}})$ 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

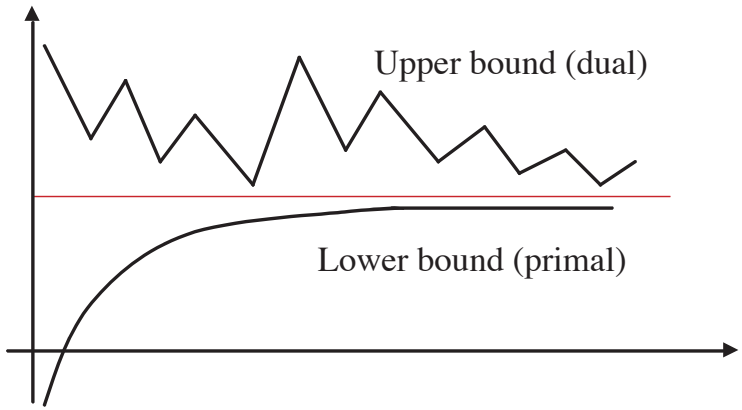
$\Longrightarrow$ tailing off, slow convergence



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 column generation

. . . 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

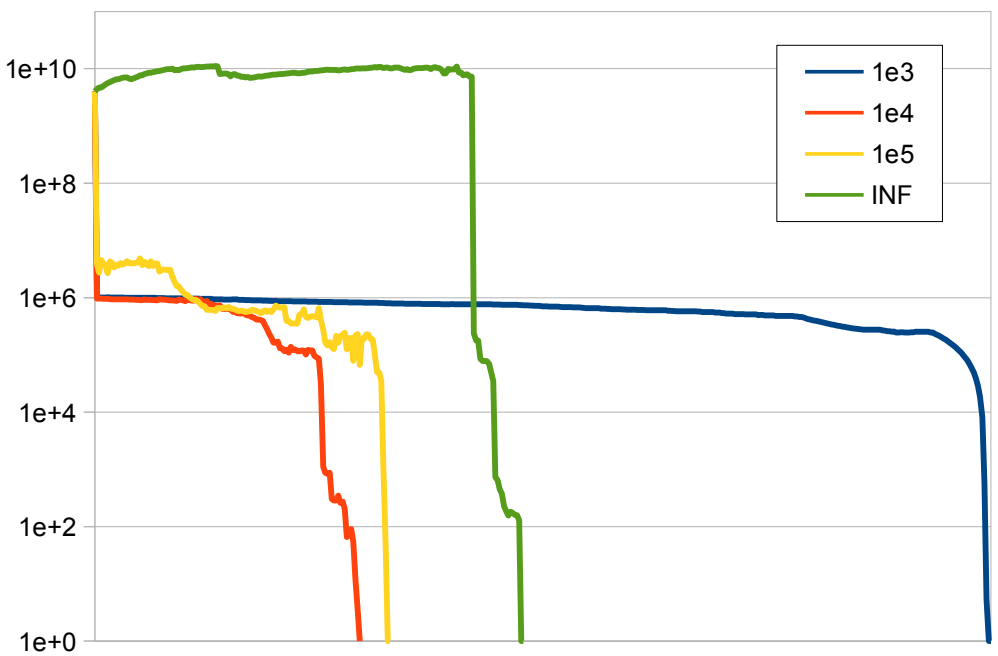
$$(\Pi_{\mathcal{B},\bar{y},t}) \quad \max \left\{ cx + \bar{y}z - t\| z \|_1 \ : \ z = b - Ax \ , \ x \in conv(\mathcal{B}) \right\} \tag{9}$$

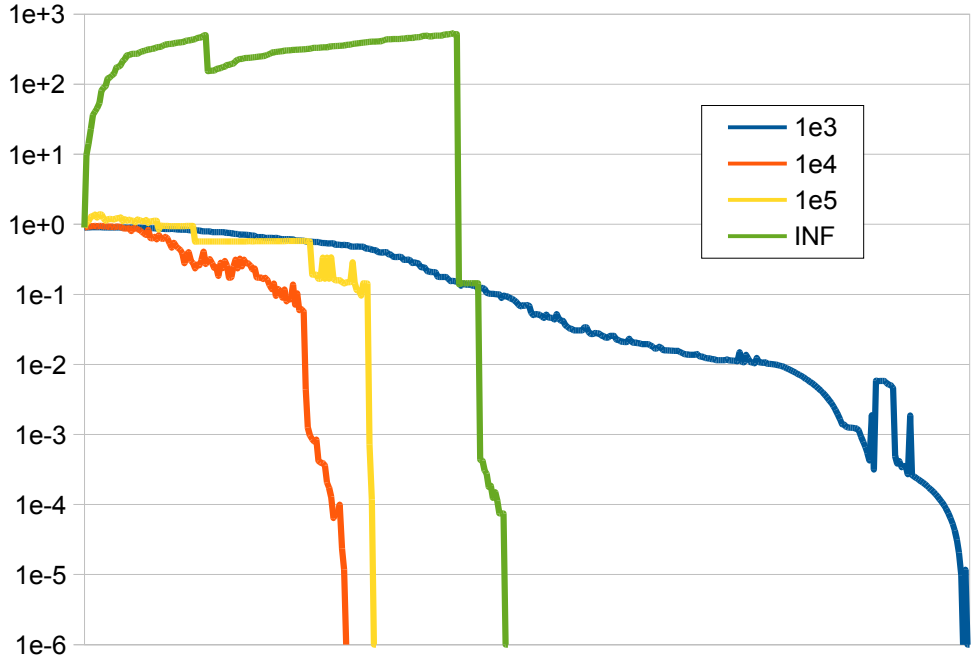
  i.e., just Dantzig-Wolfe with slacks

- When stuck and $z^* = b - Ax^* \neq 0$, either move $\bar{y}$ or enlarge t
- 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)

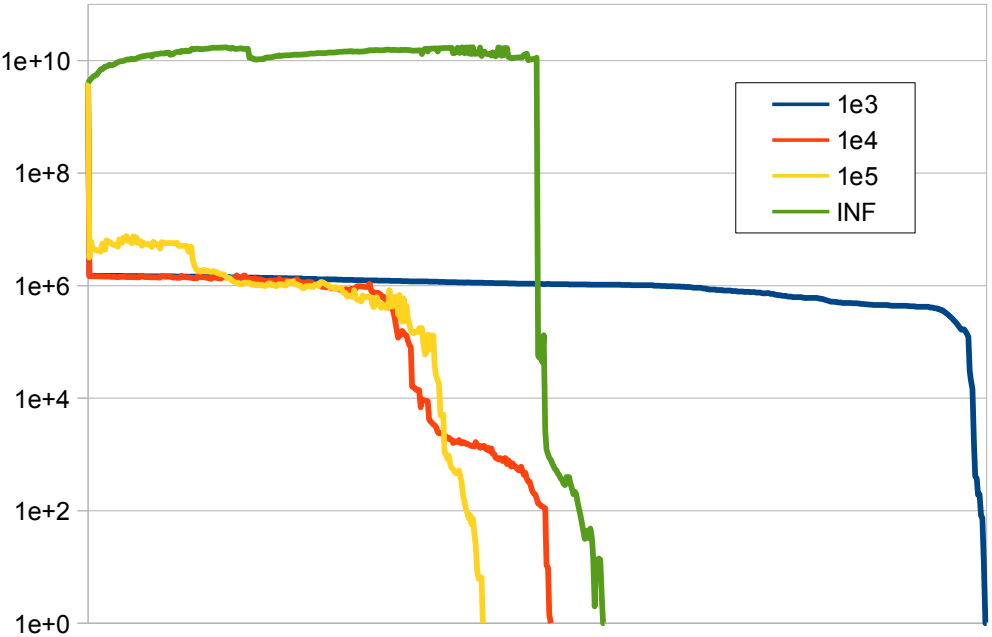


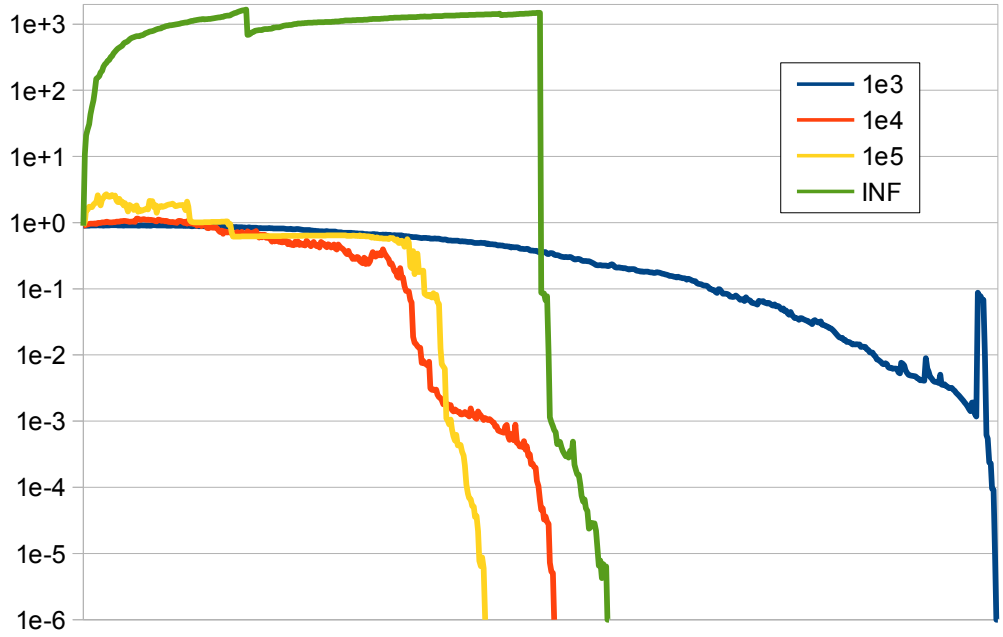


- Pure multicommodity flow instance (no y)
- Left = distance from final dual optimum

  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)

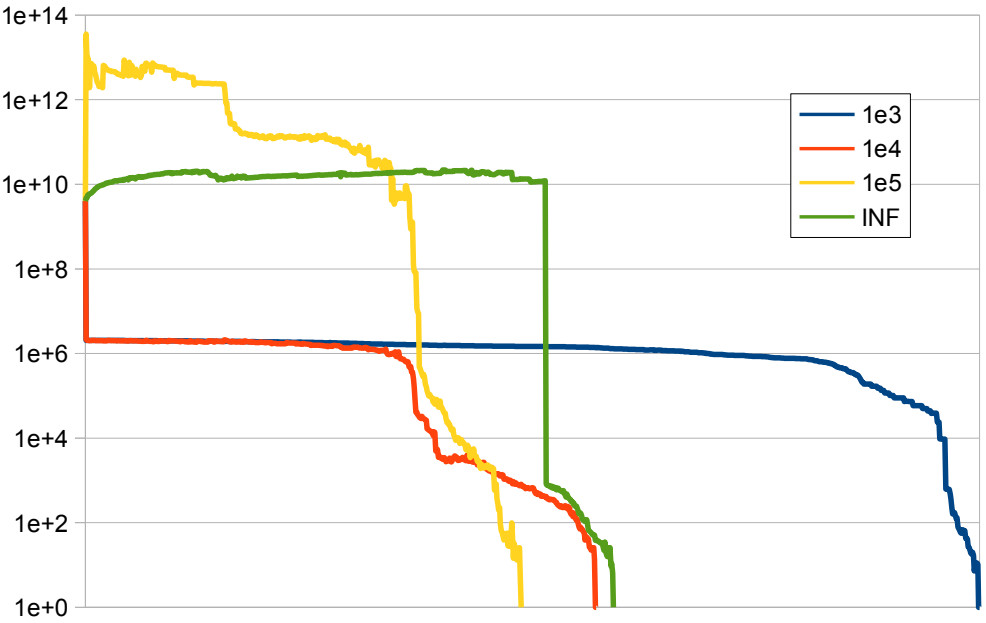


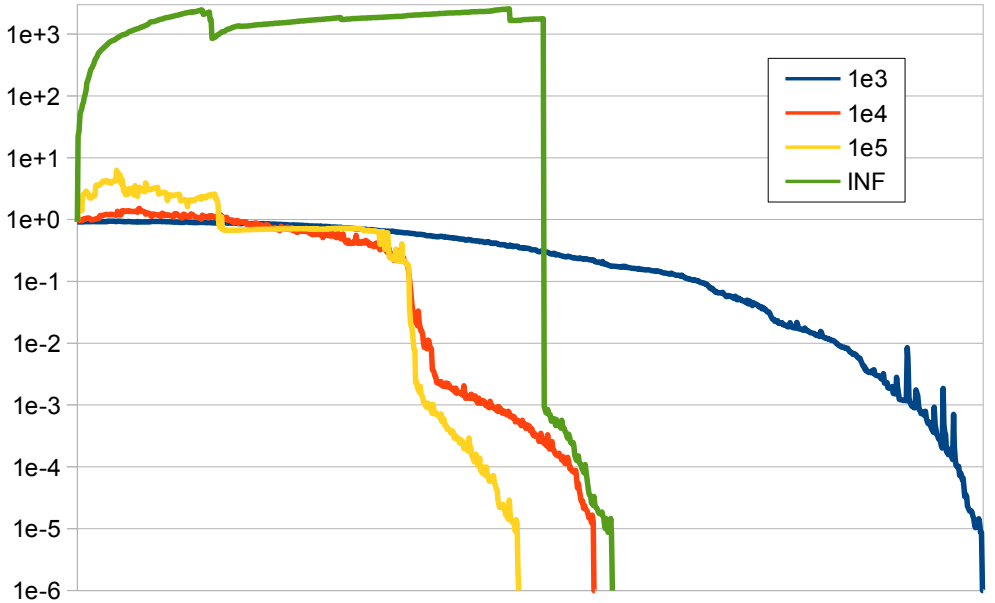


- All cases show a "combinatorial tail" where convergence is very quick
- $t =$ 1e3: "smooth but slow" until the combinatorial tail kicks in a short-step approach not unlike subgradient methods [8]
- $t = \infty$: apparently trashing along until some magic threshold is hit
- "intermediate" t work best,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

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

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

---

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

Duals, duals everywhere . . .

1. LP duality: $\max\{\, cx \,:\, Ax \leq b \,\} \;\equiv\; \min\{\, yb \,:\, yA = c\,,\, y \geq 0 \,\}$
2. (Strictly convex) QP duality: for $Q \succ 0$
$$\max\{\, cx - x^T Qx/2 \,:\, Ax \leq b \,\} \quad \equiv$$
$$\min\{\, yb + v^T Q^{-1} v/2 \,:\, yA - v = c\,,\, y \geq 0 \,\}$$
3. Conic duality: $\max\{\, cx \,:\, Ax \leq_K b \,\} \equiv \min\{\, yb \,:\, yA = c\,,\, y \geq_{K_*} 0 \,\}$
K pointed convex cone, $K_* = \{\, z : zx \geq 0 \; x \in K \,\}$ its dual
(typically the Lorentz cone $\geq_L$ or the SDP cone $\succeq$)
4. Lagrangian duality: (6) $\equiv$ (7)
5. Fenchel's duality: $\min\{\, f(y) + g(y) \,\} \;\equiv\; -\min\{\, f^*(z) + g^*(-z) \,\}$
where $f^*(y) = sup_z\{\, zy - f(y) \,\}$ is the Fenchel's conjugate of f

In general some requirement is needed: $Q \succ 0$, $A\bar{x} <_K b$ (Slater), "filling property", int dom $f \cap$ int dom $g \neq \emptyset$ or one among f, g is polyhedral, . . .

Fenchel's what???

- Fenchel's conjugate characterizes the ε-subgradients of f, i.e., the ε-subdifferential $\partial_\varepsilon f(y) = \{\, z \,:\, f(y') \geq f(y) + z(\, y' - y\,) \,\}$, as
  i) $z \in \partial_\varepsilon f(y) \iff y \in \partial_\varepsilon f^*(z) \iff f(y) + f^*(z) \leq zy + \varepsilon$

- Has many useful properties, e.g.
  ii) $f^*(0) = -\inf_y\{\, f(y)\,\}$
  iii) $f \leq g \Longrightarrow f^* \geq g^*$
  iv) $zy \leq f(y) + f^*(z) \quad \forall z, y$
  v) $(f+g)^*(z) = \inf\{\, f^*(z') + g^*(z'') \,:\, z = z' + z''\,\}$

- For a (closed) convex function $(f^*)^* = f$; for a nonconvex function, f^{**} is the (closed) convex envelope of f

- LP/QP/Conic duality offers closed formulæ for special structures

- Lagrangian duality is the general case

- Fenchel largely $\equiv$ Lagrange, often easier to use

Generalized stabilization

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

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

- With proper $\mathcal{D}$, $\phi_{\mathcal{D}}$ has same minima as f but is "smoother" (Moreau–Yosida regularization)
- Stabilized primal problem = Fenchel's dual of $(\Delta_{\bar{y},\mathcal{D}})$

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

- For our dual f, a generalized augmented Lagrangian

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

- Note: a "primal" exists even for a non-dual f:

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

$v(\Pi) = -f^*(0) = v(\Delta)$, and $f(\bar{y}) = \max\{\, \bar{y}z - f^*(z)\,\}$, i.e., the Lagrangian relaxation of (Π) w.r.t. $z = 0$

What a stabilizing term may look like?

- General properties (i) and ii) hold for $\mathcal{D} \iff$ they hold for $\mathcal{D}^*$)
  - i) $\mathcal{D} \geq 0$ convex, $\mathcal{D}(0) = 0$
  - ii) $S_\delta(\mathcal{D})$ compact and full-dimensional $\forall \delta > 0$
  - iii) $\mathcal{D}$ differentiable in 0 $\iff$ $\mathcal{D}^*$ strictly convex in 0
  - iv) $\mathcal{D}$ is "steep enough" $\implies$ $(\Delta_{\bar{y},\mathcal{D}})$ is always bounded
- Actually, iii) and iv) can be relaxed somewhat, albeit at a cost
- A slew of useful consequences (for any f, hence also $f_{\mathcal{B}}$)

$$-z^* \in \partial\mathcal{D}(d^*) \ , \ d^* \in \partial\mathcal{D}^*(-z^*) \ , \ z^* \in \partial f(\bar{x} + d^*)$$

$$\bar{y} + d^* \in \partial f^*(z^*) \ , \ \mathcal{D}(d^*) + \mathcal{D}^*(-z^*) = -z^* d^* \qquad (13)$$

$$f(\bar{y} + d^*) + f^*(z^*) = z^*(\bar{y} + d^*)$$

- Optimal solution d^*: $f(\bar{y} + d^*) < f(\bar{y})$ ($d^* = 0 \implies \bar{y}$ optimum)
- $\bar{y} := \bar{y} + d^*$, properly change $\mathcal{D}$ (or not), iterate:
  solving (Δ) by a (generalized) Proximal Point method [9]

---

[9] Rockafellar "Monotone Operators and the Proximal Point Algorithm" *SIAM J. on Control and Optimization*, 1975

Approximated generalized stabilization

- All nice and well, except $(\Delta_{\bar{y},\mathcal{D}})$ is as difficult to solve as (Δ)
- However, we can solve $(\Delta_{\bar{y},\mathcal{D}})$ by column generation
- Stabilized master problems

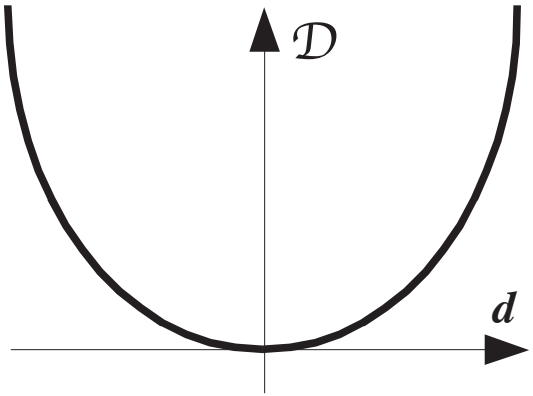
$$\begin{aligned}(\Delta_{\mathcal{B},\bar{y},\mathcal{D}})\ & \min\left\{ f_{\mathcal{B}}(\,\bar{y}+d\,)+\mathcal{D}(\,d\,)\right\}\\ (\Pi_{\mathcal{B},\bar{y},\mathcal{D}})\ & \max\left\{ cx+\bar{y}(b-Ax)-\mathcal{D}^*(Ax-b)\ :\ x\in conv(\mathcal{B})\right\}\end{aligned} \tag{14}$$

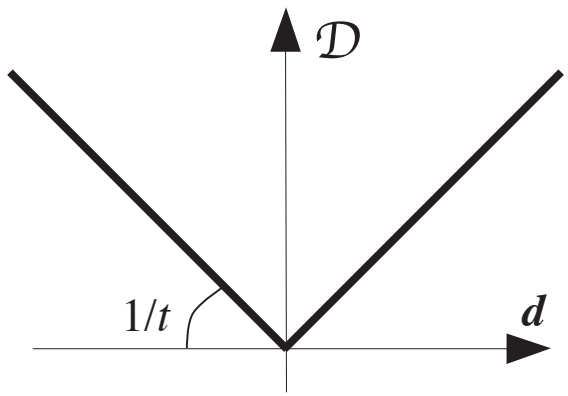
- Evaluate $f(\bar{y}+d^*)$, update $\mathcal{B}$, iterate $\Longrightarrow$ solve $(\Delta_{\bar{y},\mathcal{D}})$ / $(\Pi_{\bar{y},\mathcal{D}})$
- Remind: d^* has to ensure descent of f, but we compute $f(\bar{y}+d^*)$ $\Longrightarrow$ early termination: stop as soon as $f(\bar{y}+d^*) \ll f(\bar{y})$
- Overall convergent method for $(\Delta)/(\Pi)$ [10], particularly nifty trick: aggregation. With proper $\mathcal{D}$ $(S_0(\mathcal{D})=\{0\})$ $\mathcal{B}=\{\,x^*\,\}$ converges (rather slowly [11]: "poorman bundle" $\approx$ volume [12] $\equiv$ subgradient)

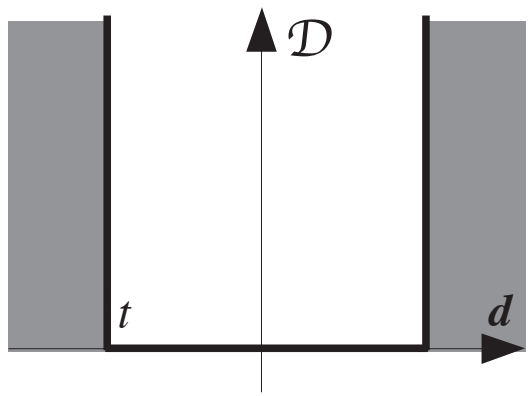
---

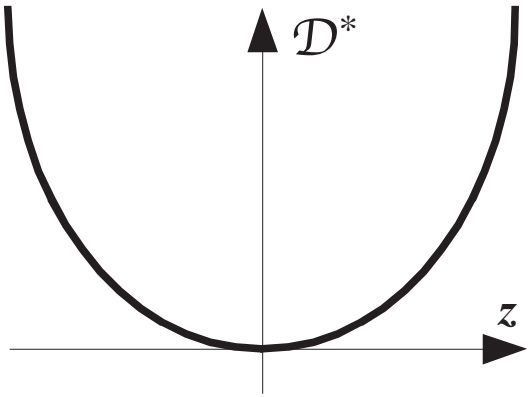
[10] F. "Generalized Bundle Methods" *SIAM Journal on Optimization* 2002
[11] Briant, Lemaréchal, et. al. "Comparison of bundle and classical column generation" *Mathematical Programming* 2006
[12] Bahiense, Maculan, Sagastizábal "The volume algorithm revisited: relation with bundle methods" *Math. Prog.* 2002

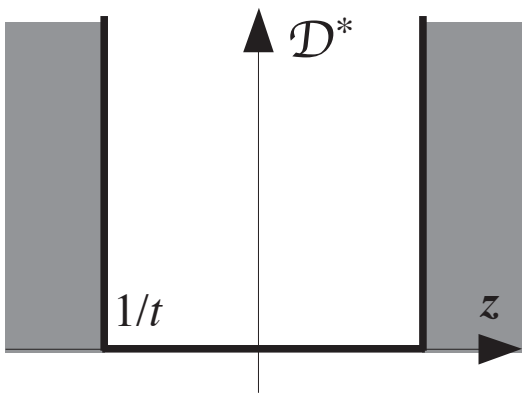
Classical stabilizing terms

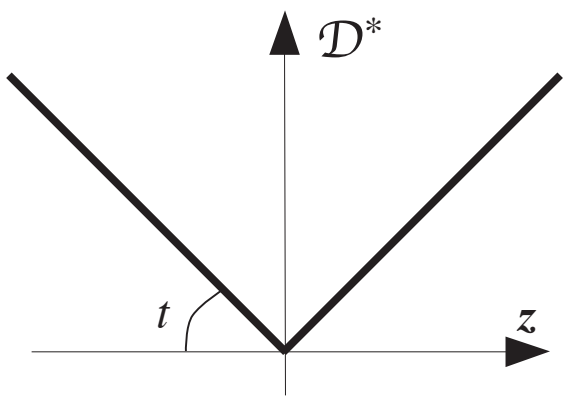












[8] $\mathcal{D} = \frac{1}{2t}\|\cdot\|_2^2$, $\mathcal{D}^* = \frac{1}{2}t\|\cdot\|_2^2$

[13] $\mathcal{D} = \frac{1}{t}\|\cdot\|_1$, $\mathcal{D}^* = I_{B_\infty(1/t)}$

[7] $\mathcal{D} = I_{B_\infty(t)}$, $\mathcal{D}^* = t\|\cdot\|_1$

[13] Kim, Chang, Lee "A descent method with LP subproblems for nondifferentiable convex optimization" *Math. Prog.* 1995

Fancier stabilizing terms

- Smooth approximation of $\|\cdot\|_1$ [14]

$$\mathcal{D}^*(z) = \sum_i \Phi^*_\varepsilon(z_i) = \begin{cases} z_i^2/(2\varepsilon) & \text{if } -\varepsilon \le z_i \le \varepsilon \\ |z_i| - \frac{\varepsilon}{2} & \text{otherwise} \end{cases} \tag{15}$$

- Smooth approximation of $t\|\cdot\|_\infty$ [15]

$$\mathcal{D}^*(z) = \ln \sum_i e^{tz_i} \tag{16}$$

- Bregman functions [16]

$$\mathcal{D}_{\bar{y}}(d) = (\,\psi(\bar{y}+d) - \psi(\bar{y}) - \nabla\psi(\bar{y})d\,) \tag{17}$$

  with ψ fixed, strictly convex, differentiable, with compact level sets

- Others (φ-divergences, ...)
- However, all these are "very nonlinear" which is a serious issue

---

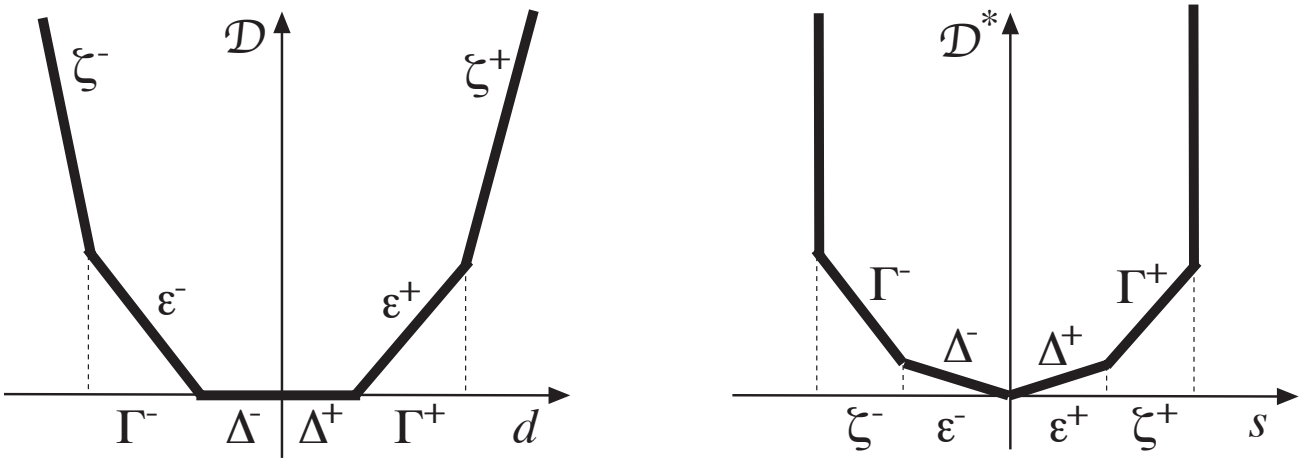
[14] Pinar, Zenios "Parallel decomposition of multicommodity network flows using a linear-quadratic ..." *ORSA J. Comp.* 1992

[15] Grigoriadis, Kahchiyan "An exponential-function reduction method for block-angular convex programs" *Networks* 1995

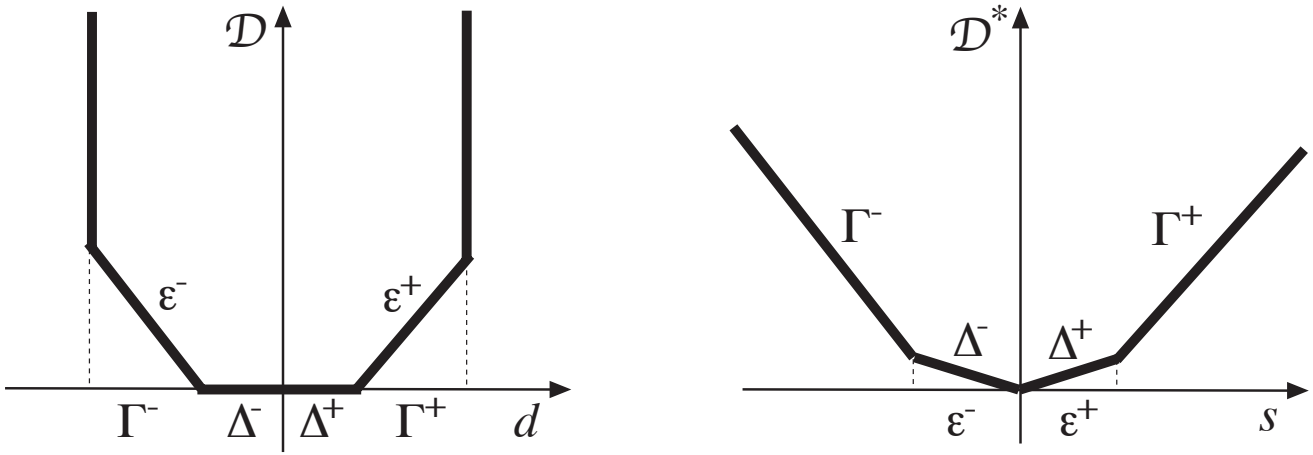
[16] Chen, Teboulle "Convergence analysis of a proximal-like minimization algorithm using Bregman functions" *SIOPT* 1993

A 5-piecewise-linear function

Trust region on $\bar{y}$ + small penalty close + much larger penalty farther [17]



Slightly simplified version: only 3 pieces.



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

A 5-piecewise-linear function (cont.d)

$\mathcal{D}(u) = \sum_{i=1}^{m} \mathcal{D}_i(u_i)$ where

$$\mathcal{D}_i(u_i) = \begin{cases} -(\zeta_i^- + \varepsilon_i^-)(u_i + \Gamma_i^-) - \zeta_i^- \Delta_i^- & \text{if} \quad -\infty \le u_i \le -\Gamma_i^- - \Delta_i^- \\ -\varepsilon_i^-(u_i - \Delta_i^-) & \text{if} \quad -\Gamma_i^- - \Delta_i^- \le u_i \le -\Delta_i^- \\ 0 & \text{if} \quad -\Delta_i^- \le u_i \le \Delta_i^+ \\ +\varepsilon_i^+(u_i - \Delta_i^+) & \text{if} \quad \Delta_i^+ \le u_i \le \Delta_i^+ + \Gamma_i^+ \\ +(\varepsilon_i^+ + \zeta_i^+)(u_i - \Gamma_i^+) - \zeta_i^+ \Delta_i^+ & \text{if} \quad \Delta_i^+ + \Gamma_i^+ \le u_i \le +\infty \end{cases}$$

$\mathcal{D}^*(z) = \sum_{i=1}^{m} \mathcal{D}_i^*(z_i)$ where

$$\mathcal{D}_i^*(z_i) = \begin{cases} +\infty & \text{if} \quad z_i < -(\zeta_i^- + \varepsilon_i^-) \\ -(\Gamma_i^- + \Delta_i^-)z_i - \Gamma_i^- \varepsilon_i^- & \text{if} \quad -\zeta_i^- - \varepsilon_i^- \le z_i \le -\varepsilon_i^- \\ -\Delta_i^- z_i & \text{if} \quad -\varepsilon_i^- \le z_i \le 0 \\ +\Delta_i^+ z_i & \text{if} \quad 0 \le z_i \le \varepsilon_i^- \\ +(\Gamma_i^+ + \Delta_i^+)z_i - \Gamma_i^+ \varepsilon_i^+ & \text{if} \quad \varepsilon_i^+ \le z_i \le (\zeta_i^+ + \varepsilon_i^+) \\ +\infty & \text{if} \quad z_i > (\zeta_i^+ + \varepsilon_i^+) \end{cases}$$

Interval widths $\Longleftrightarrow$ penalties

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
- $(c - yA)r > 0 \Rightarrow f(y) = +\infty \Rightarrow$ constraint $cr \leq y(Ar)$ in dual space

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

  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
- $(c - yA)r > 0 \Rightarrow f(y) = +\infty \Rightarrow$ constraint $cr \leq y(Ar)$ in dual space

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

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
- $(c - yA)r > 0 \Rightarrow f(y) = +\infty \Rightarrow$ constraint $cr \leq y(Ar)$ in dual space

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

  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

Disaggregated Model

Dantzig-Wolfe and Multicommodity flows

- Dantzig-Wolfe reformulation
- $\mathcal{S} = \{$ (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 $\theta_s^k = \theta_s^h, h \neq k$)

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, $\mathcal{P} = \cup_{k \in K} \mathcal{P}^k$

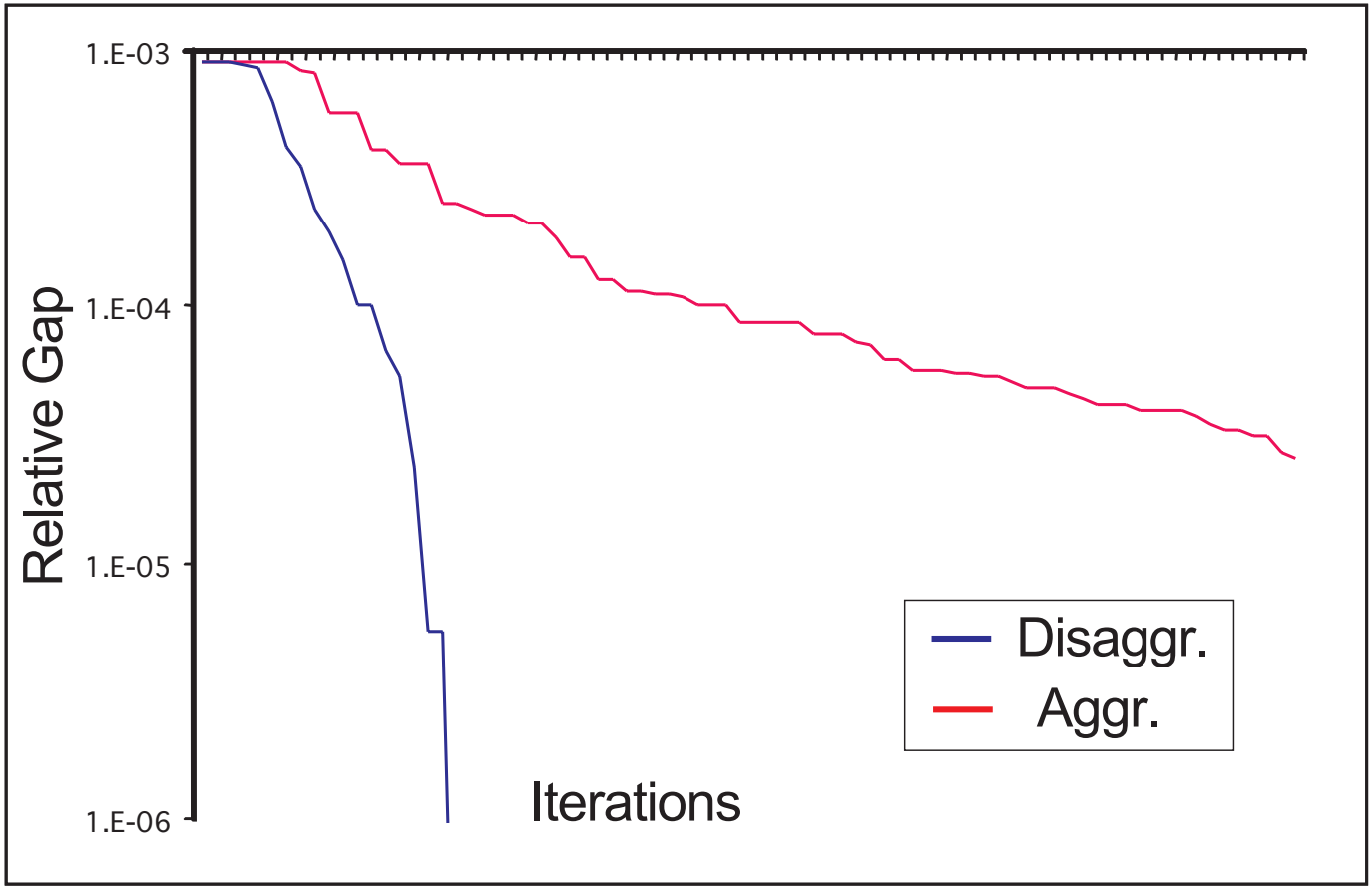
$$
\begin{array}{lll}
\min & \sum_{p \in \mathcal{P}} c_p f_p & \\
& \sum_{p \in \mathcal{P} \,:\, (i,j) \in p} f_p \leq u_{ij} & (i,j) \in A \\
& \sum_{p \in \mathcal{P}^k} f_p = d^k & k \in K \\
& f_p \geq 0 & p \in \mathcal{P}
\end{array}
$$

- Disaggregated formulation: more columns but sparser, more rows
- Much more efficient than aggregated formulation [18]
- Master problem size $\approx$ time increases, but convergence speed increases a lot $\equiv$ consistent improvement

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

Disaggregated decomposition

- Easily extended to any decomposable X [19]
- Stabilized versions immediate



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

Dual-Optimal Cuts

Dual-Optimal Cuts

- Stabilizing = restricting the dual space
- The above approaches need stability center $\bar{y}$, to be updated: it'd be nice if we could do without
- Simple observation: dual constraints = primal variables
  $\Longrightarrow$ need to add even more variables to the primal

  . . . in such a way that not all dual optimal solution are cut

Dual-Optimal Cuts

- Stabilizing = restricting the dual space
- The above approaches need stability center $\bar{y}$, to be updated: it'd be nice if we could do without
- Simple observation: dual constraints = primal variables
  $\Longrightarrow$ need to add even more variables to the primal

  . . . in such a way that not all dual optimal solution are cut
- Actually quite simple:
  the new variables must not add new primal solutions [20]

---

[20] Ben Amor, Desrosiers, Valério de Carvalho "Dual-optimal inequalities for stabilized column generation" *Op. Res.* 2006

Dual-Optimal Cuts for Multicommodity flows

- $\mathcal{C}$ = directed circuits with one reversed arc (aggregated flow)
- Constraints become

$$\sum_{p \in \mathcal{P} \,:\, (i,j) \in p} f_p + \sum_{c \in \mathcal{C} \,:\, (i,j) \in c} \pm f_c \leq u_{ij}$$

  where "$-$" if (i,j) is reversed in c; hence, one also needs

$$0 \leq \sum_{p \in \mathcal{P} \,:\, (i,j) \in p} f_p + \sum_{c \in \mathcal{C} \,:\, (i,j) \in c} \pm f_c$$

- Any feasible solution to the extended model can be converted into a feasible solution to the original model

Dual-Optimal Cuts for Multicommodity flows

- $\mathcal{C}$ = directed circuits with one reversed arc (aggregated flow)
- Constraints become

$$\sum_{p \in \mathcal{P} : (i,j) \in p} f_p + \sum_{c \in \mathcal{C} : (i,j) \in c} \pm f_c \leq u_{ij}$$

  where "$-$" if (i,j) is reversed in c; hence, one also needs

$$0 \leq \sum_{p \in \mathcal{P} : (i,j) \in p} f_p + \sum_{c \in \mathcal{C} : (i,j) \in c} \pm f_c$$

- Any feasible solution to the extended model can be converted into a feasible solution to the original model
- $|\mathcal{C}| \in O(n^2)$ if G is planar, all-pairs SPT pricing otherwise
- Promising initial results
- Other applications: Cutting Stock (different cuts)

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 [21]

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

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

- . . . and with very long, dense rows

[21] 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 [22]

---

[22] 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

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

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"

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

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

$$(\Pi_{\mathcal{B},\bar{y},\mathcal{D}}) \quad \max \begin{cases} c_1 \left(\sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta_{\bar{x}_1} \right) + c_2(x_2) + \bar{y} z - \mathcal{D}^*(-z) \\ z = b - A_1 \left(\sum_{\bar{x}_1 \in \mathcal{B}} \bar{x}_1 \theta_{\bar{x}_1} \right) - A_2 x_2 \\ \sum_{\bar{x}_1 \in \mathcal{B}} \theta_{\bar{x}_1} = 1 \quad , \quad G(x_2) \leq g \end{cases} \tag{18}$$

- 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 [23])
- 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 [24,25] 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

[23] F. "Solving semidefinite quadratic problems within nonsmooth optimization algorithms" *Computers & O.R.* 1996
[24] F., Lodi, Rinaldi "New approaches for optimizing over the semimetric polytope" *Mathematical Programming* 2005
[25] 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 [26])

[26] 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:

$$\min \left\{ \sum_{(i,j)\in A} \frac{y_{ij}}{1-y_{ij}} \ : \ (1)-(4)\ ,\ y \in [0,1]^{|A|} \right\}$$

  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 [27]

- Substantially improved performances

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

Structured Decomposition

Knapsack decomposition for the general model

- Relax the flow conservation constraints (2)

$$\begin{aligned}
\min \quad & \textstyle\sum_{(i,j)\in A} \big(\sum_{k\in K} (d^k c^k_{ij} - \pi^k_i + \pi^k_j) x^k_{ij} + f_{ij} y_{ij} \big) & \\
& \textstyle\sum_{k\in K} d^k x^k_{ij} \le u_{ij} y_{ij} & (i,j) \in A \\
& x^k_{ij} \in [0,1] & (i,j) \in A,\ k \in K \\
& y_{ij} \in \mathbb{N} & (i,j) \in A
\end{aligned}$$

- Decomposes by arc, easy ($\approx$ 2 continuous knapsack) but no integrality property $\Longrightarrow$ better bound than continuous relaxation
- Residual capacity inequalities, separate $\approx$ 2 continuous knapsack [28]

$$\begin{gathered}
a_k = d^k / u_{ij} \qquad\qquad a(S) = \textstyle\sum_{k\in S} a_k \qquad\qquad S \subseteq K \\
\textstyle\sum_{k\in S} a_k (1 - x^k_{ij}) \ge \big(a(S) - \lfloor a(S) \rfloor \big) \big(\lceil a(S) \rceil - y \big)
\end{gathered} \tag{19}$$

- $\bar{I}+$ = continuous relaxation of (1)–(5) + (19) $\equiv$ DW [29]

---

[28] Atamtürk "On Capacitated Network Design Cut-Set Polyhedra" *Mathematical Programming* 2002

[29] Magnanti, Mirchandani, Vachani "The Convex Hull of Two Core Capacitated Network Design Problems" *Math. Prog.* 1993

Computational results: RG vs. StabDW

- Intel Xeon X7350@2.93GHz, 64Gb RAM, Suse Linux, `CPLEX 11.1`
- Large-scale instances ($|K| \in \{100, 200, 400\}$), very difficult
- $C = 1 \Rightarrow$ lightly capacitated, $C = 16 \Rightarrow$ tightly capacitated

Computational results: RG vs. StabDW

- Intel Xeon X7350@2.93GHz, 64Gb RAM, Suse Linux, `CPLEX 11.1`
- Large-scale instances ($|K| \in \{100, 200, 400\}$), very difficult
- $C = 1 \Rightarrow$ lightly capacitated, $C = 16 \Rightarrow$ tightly capacitated
- DW unbearably slow, disaggregating does not help (enough)
- Stabilized DW much better, but only if disaggreated
- $\|\cdot\|_\infty$ stabilization (QP too costly, see below)

Sample computational results ($|K| = 100$)

Problem			$l+$		StabDW	
$\|A\|$	C	imp	cpu	it	cpu	it
517	1	187.00	348	26	4323	88144
	4	138.22	362	25	3581	79390
	8	100.08	305	21	4054	88807
	16	60.49	249	21	3015	71651
517	1	155.19	140	23	2899	69500
	4	122.84	194	26	2799	65229
	8	93.00	151	20	2824	66025
	16	59.68	116	18	2172	56184
669	1	114.50	80	26	330	11273
	4	97.32	78	22	327	10951
	8	79.62	68	19	323	11173
	16	56.19	58	19	275	9979

- RG always better than StabCG

Sample computational results ($|K| = 200$)

Problem			$l+$		StabDW	
$\|A\|$	C	imp	cpu	it	cpu	it
229	1	205.67	49081	109	11748	154821
	4	131.24	30899	91	9132	131674
	8	84.61	16502	87	12682	162766
	16	42.78	2090	54	6541	97952
229	1	185.17	18326	86	9261	132963
	4	125.39	15537	80	11791	147879
	8	85.31	9500	74	10702	146727
	16	46.09	1900	52	7268	107197
287	1	198.87	14559	66	8815	120614
	4	136.97	11934	62	8426	112308
	8	92.94	9656	64	10098	130536
	16	53.45	3579	54	6801	98972

- RG wins only for large C, basically both lose

Reformulation III: Binary formulation B

- Redundant upper bound constraints: $y_{ij} \leq \left\lceil \sum_{k \in K} d^k / a_{ij} \right\rceil = T_{ij}$
- Pseudo-polinomially many segments $S_{ij} = \{ 1, \ldots, T_{ij} \}$ for y_{ij}

Reformulation III: Binary formulation B

- Redundant upper bound constraints: $y_{ij} \leq \left\lceil \sum_{k \in K} d^k / a_{ij} \right\rceil = T_{ij}$
- Pseudo-polinomially many segments $S_{ij} = \{ 1, \ldots, T_{ij} \}$ for y_{ij}
- Reformulation in binary variables: $y_{ij} = \sum_{s \in S_{ij}} y_{ij}^s$

$$y_{ij}^s = \begin{cases} 1 & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \qquad s \in S_{ij}$$

$$u_{ij}^{ks} = \begin{cases} u_{ij}^k & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \qquad s \in S_{ij} \ , \ k \in K$$

$$(s-1) a_{ij} y_{ij}^s \leq \sum_{k \in K} d^k u_{ij}^{ks} \leq s a_{ij} y_{ij}^s \qquad (i,j) \in A \ , \ s \in S_{ij}$$

$$\sum_{s \in S_{ij}} y_{ij}^s \leq 1 \qquad (i,j) \in A$$

... then original variables can be removed

- Up to now, continuous relaxation bound has not improved

Improved binary formulation $B+$

- Extended linking inequalities:

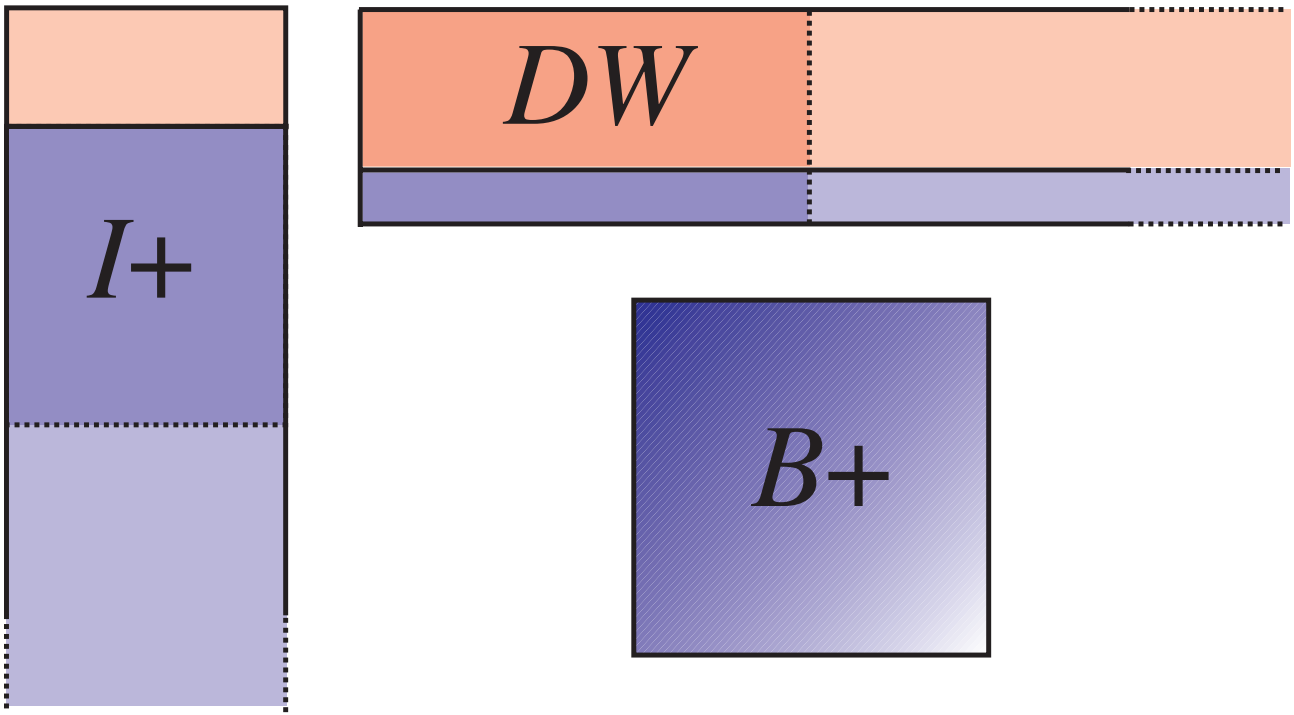
$$u_{ij}^{ks} \leq y_{ij}^{s} \qquad (i,j) \in A \ , \ k \in K \ , \ s \in S_{ij}$$

- Improved relaxation bound: $v(\bar{B}+) = v(\bar{I}+) = v(DW)$ [30]
- In particular, binary formulation describes $conv(X^{ij})$: continuous relaxation has integrality property
- Optimizing over $X \Longrightarrow conv(X)$ easy
- Pseudo-polynomial number of variables and constraints
- How can we exploit it?

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

The main issue

- Substantially different from both RG and DW



- Need to generate both rows and columns

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

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

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

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

- Assumption 2: padding with zeroes

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

$$\Rightarrow X_{\mathcal{B}} = \left\{ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ : \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \subseteq conv(X)$$

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

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

- Assumption 2: padding with zeroes

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

$$\Rightarrow X_{\mathcal{B}} = \left\{ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ : \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \subseteq conv(X)$$

- Assumption 3: easy update of rows and columns

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

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

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

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

- Assumption 2: padding with zeroes

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

$$\Rightarrow X_{\mathcal{B}} = \left\{ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ : \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \subseteq conv(X)$$

- Assumption 3: easy update of rows and columns

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

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

- Structured master problem

$$(\Pi_{\mathcal{B}}) \qquad \max\left\{ cx \ : \ Ax = b \ , \ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \tag{20}$$

$\equiv$ structured model

$$f_{\mathcal{B}}(y) \ = \ \max\{ \ (c - yA)x + xb \ : \ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \ \} \tag{21}$$

The Structured Dantzig-Wolfe Algorithm

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

The Structured Dantzig-Wolfe Algorithm

$\langle$ 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 $v^* < c\bar{x} + y^*(b - A\bar{x})$

- Relatively easy [30] to prove that:
  - finitely terminates with an optimal solution of (Π)
  - . . . even if (proper) removal from $\mathcal{B}$ is allowed (when cx^* increases)
  - . . . even if X is non compact and $\mathcal{B} = \emptyset$ at start (Phase 0)

The Structured Dantzig-Wolfe Algorithm

$\langle$ 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 $v^* < c\bar{x} + y^*(b - A\bar{x})$

- Relatively easy [30] to prove that:
  - finitely terminates with an optimal solution of (Π)
  - ...even if (proper) removal from $\mathcal{B}$ is allowed (when cx^* increases)
  - ...even if 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

Computational results for StructDW

- Same machine/instances as before
- Solving the root relaxation, then freezing the formulation
  + CPLEX polishing for one hour
- Unlike $I+$, frozen $B+$ formulations may not contain optimal solution
  $\Longrightarrow$ final gap $\approx$ quality of obtained formulation
- imp = lower bound improvement w.r.t. $\bar{I}$ (= for all)
  gap = final gap (%), cpu = time, it = iterations

Sample computational results ($|K| = 100$)

Problem			$I+$			StabDW		StructDW		
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it
517	1	187.00	348	5.78	26	4323	88144	296	6.94	55
	4	138.22	362	6.42	25	3581	79390	312	7.48	44
	8	100.08	305	6.12	21	4054	88807	633	6.11	61
	16	60.49	249	6.20	21	3015	71651	1138	6.45	87
517	1	155.19	140	3.95	23	2899	69500	188	4.70	60
	4	122.84	194	3.87	26	2799	65229	147	4.15	39
	8	93.00	151	3.96	20	2824	66025	355	4.31	67
	16	59.68	116	4.72	18	2172	56184	551	4.94	70
669	1	114.50	80	0.50	26	330	11273	36	0.46	32
	4	97.32	78	0.46	22	327	10951	66	0.46	50
	8	79.62	68	0.46	19	323	11173	55	0.46	33
	16	56.19	58	0.74	19	275	9979	164	0.81	65

- SDW worsens as C grows (tighter capacities), RG the converse

Sample computational results ($|K| = 200$)

Problem			$I+$			StabDW		StructDW		
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it
229	1	205.67	49081	28.16	109	11748	154821	525	10.50	44
	4	131.24	30899	25.40	91	9132	131674	807	13.58	45
	8	84.61	16502	21.80	87	12682	162766	1593	10.17	44
	16	42.78	2090	5.59	54	6541	97952	2630	9.20	73
229	1	185.17	18326	20.53	86	9261	132963	380	7.44	39
	4	125.39	15537	18.81	80	11791	147879	612	9.36	49
	8	85.31	9500	13.08	74	10702	146727	1647	8.87	68
	16	46.09	1900	7.19	52	7268	107197	3167	7.99	108
287	1	198.87	14559	27.86	66	8815	120614	598	12.54	53
	4	136.97	11934	22.52	62	8426	112308	603	15.07	37
	8	92.94	9656	15.28	64	10098	130536	1221	10.38	41
	16	53.45	3579	11.60	54	6801	98972	3515	9.06	99

- Same trend, but RG better only for $C = 16$

Sample computational results ($|K| = 400$)

Problem			StabDW		StructDW		
$\|A\|$	C	imp	cpu	it	cpu	gap	it
519	1	100.83	87695	248746	9839	9.96	157
	4	92.54	88031	247864	9087	11.25	140
	8	82.16	88918	258266	11613	8.47	143
	16	65.53	85384	238945	38617	10.26	242
519	1	125.07	93065	258054	22246	14.90	165
	4	111.02	90573	250854	17976	18.22	131
	8	94.82	93418	256884	30460	18.18	159
	16	71.31	93567	265663	74447	16.50	176
668	1	126.02	98789	246702	23771	11.89	149
	4	115.29	99014	247620	28567	10.97	176
	8	102.03	104481	258636	27871	12.07	130
	16	80.96	103011	278905	58363	13.95	156

- SWD always better, stabilizing SDW seems promising

Stabilizing the Structured Dantzig-Wolfe Algorithm

- Exactly the same as stabilizing DW: stabilized master problem

$$(\Delta_{\mathcal{B},\bar{y},\mathcal{D}}) \qquad \min \left\{ f_{\mathcal{B}}(\bar{y} + d) + \mathcal{D}(d) \right\} \tag{22}$$

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

- Even simpler from the primal viewpoint [31]:

$$\max \left\{ cx + \bar{y}z - \mathcal{D}^*(-z) \,:\, z = b - Ax \,,\, x = C_{\mathcal{B}}\theta_{\mathcal{B}} \,,\, \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \tag{23}$$

- With proper choice of $\mathcal{D}$, still a Linear Program; e.g.

$$\begin{aligned} \max \quad & \ldots - (\Delta^- + \Gamma^-)z_2^- - \Delta^- z_1^- - \Delta^+ z_1^+ - (\Delta^+ + \Gamma^+)z_2^+ \\ & z_2^- + z_1^- - z_1^+ - z_2^+ = b - Ax \;\;,\;\; \ldots \\ & z_2^+ \geq 0 \;\;,\;\; \varepsilon^+ \geq z_1^+ \geq 0 \;\;,\;\; \varepsilon^- \geq z_1^- \geq 0 \;\;,\;\; z_2^- \geq 0 \end{aligned}$$

- Dual optimal variables of "$z = b - Ax$" 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

---

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

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{ x^* \}$ ($\mathcal{B} = \{ x^* \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{ x^* \}$ ($\mathcal{B} = \{ x^* \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible ... or is it? Actually, not!
- $\bar{f}_{\mathcal{B}} = \max\{ f_{\mathcal{B}} , f_{x^*}(y) = cx^* + y(b - Ax^*) \}$ is a model of f

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{ x^* \}$ ($\mathcal{B} = \{ x^* \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible ... or is it? Actually, not!
- $\bar{f}_{\mathcal{B}} = \max\{ f_{\mathcal{B}} , f_{x^*}(y) = cx^* + y(b - Ax^*) \}$ is a model of f
- Stabilized master problem with $\bar{f}_{\mathcal{B}}$

$$\max \begin{cases} cx + (1-\rho)cx^* + \bar{y}z - \mathcal{D}^*(-z) \\ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ z = Ax + (1-\rho)Ax^* - b \ , \ \ \rho \in [0,1] \end{cases} \tag{24}$$

  if $conv(X)$ compact and constraints linear
- "Knob": $\rho = 0 \ \Rightarrow \ \gamma_{\mathcal{B}} = 0 \ \Rightarrow \ x = x^*$, $\rho = 1 \ \Rightarrow \ x \in X_{\mathcal{B}}$

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{ x^* \}$ ($\mathcal{B} = \{ x^* \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible ... or is it? Actually, not!
- $\bar{f}_{\mathcal{B}} = \max\{ f_{\mathcal{B}} , f_{x^*}(y) = cx^* + y(b - Ax^*) \}$ is a model of f
- Stabilized master problem with $\bar{f}_{\mathcal{B}}$

$$\max \begin{cases} cx + (1-\rho)cx^* + \bar{y}z - \mathcal{D}^*(-z) \\ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ z = Ax + (1-\rho)Ax^* - b \ , \ \rho \in [0,1] \end{cases} \tag{24}$$

  if $conv(X)$ compact and constraints linear

- "Knob": $\rho = 0 \Rightarrow \gamma_{\mathcal{B}} = 0 \Rightarrow x = x^*$, $\rho = 1 \Rightarrow x \in X_{\mathcal{B}}$
- Possible use: avoid Phase 0 when $\mathcal{D}$ "not steep"
  given $x^* \in conv(X)$ (e.g. $x^* \in X$) such that $Ax^* = b$

Computational results

- Same machine/instances as before
- Comparing SDW with S^{2}DW
- No removal/aggregation for $\mathcal{B}$, fixed t (class-specific tuning)
- Different stabilizing terms: $\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2$ vs $\mathcal{D}_t = I_{B_\infty(t)}$ (QP vs LP, Lemaréchal vs Marsten)
- Different warm-start: "standard" MCF initialization (used for all) vs MCF + subgradient warm-start (few iterations, class-specific tuning)
- gap = final gap (%), cpu = time, it = iterations, ss = serious steps
- Master problem time $>$ 90%, very possibly 99%

Sample computational results ($|K| = 100$)

	StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
C	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
1	296	6.94	55	16380	6.57	51	15	223	2.97	66	58	357	1.52	91	84
4	312	7.48	44	17091	5.87	47	12	298	2.72	70	54	270	1.48	69	60
8	633	6.11	61	22176	7.16	37	14	280	2.70	64	34	277	1.44	65	47
16	1138	6.45	87	27033	6.08	43	18	190	2.78	60	21	119	1.52	40	18
1	188	4.70	60	5802	4.01	42	13	205	2.56	71	57	222	1.43	85	71
4	147	4.15	39	6453	4.32	39	15	215	2.43	79	40	91	1.39	41	36
8	354	4.31	67	5752	4.40	31	12	167	2.38	62	25	124	1.42	50	21
16	551	4.94	70	10154	5.07	40	14	163	2.76	61	20	113	1.53	50	19
1	36	0.46	32	2405	0.46	47	15	84	0.41	76	48	78	0.33	72	66
4	66	0.46	50	1964	0.46	45	14	67	0.41	74	24	81	0.33	73	56
8	55	0.46	33	1974	0.46	44	15	50	0.41	57	18	40	0.33	49	20
16	164	0.81	65	1408	0.80	38	17	47	0.61	52	16	44	0.40	52	22

- S^2DW_2 converges faster but slow, ws^2 best in gap and often time

Sample computational results ($|K| = 200$)

	StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
C	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
1	525	10.50	44	1.8e4	12.11	32	17	860	4.16	76	73	907	1.32	129	119
4	807	13.58	45	2.7e4	10.20	29	15	1091	2.79	89	87	1460	1.23	126	118
8	1593	10.17	44	8.3e4	10.12	40	17	1027	3.03	78	61	1237	1.20	99	77
16	2630	9.20	73	1.1e5	9.21	54	16	399	2.12	65	31	804	1.02	114	73
1	380	7.44	39	1.0e4	****	29	14	557	2.61	80	71	592	1.30	101	95
4	612	9.36	49	1.3e4	10.33	25	15	755	2.87	80	68	930	1.22	98	95
8	1647	8.87	68	3.3e4	10.61	30	14	468	2.75	50	43	761	1.33	83	66
16	3167	7.99	108	7.0e4	8.32	47	17	476	2.22	67	30	357	1.10	53	39
1	598	12.54	53	2.1e4	16.31	39	15	1019	3.92	98	93	1327	1.65	149	143
4	603	15.07	37	1.8e4	13.78	27	15	1001	3.72	90	79	891	1.60	98	94
8	1221	10.38	41	5.2e4	11.81	29	14	909	3.68	73	50	1040	1.63	102	96
16	3515	9.06	99	1.3e5	10.11	54	17	513	2.93	59	25	555	1.26	62	45

- S^2DW_2 exceedingly slow, ws^2 best in gap, not always time

Sample computational results ($|K| = 400$)

	StructDW			S^2DW_∞				S^2DW_∞–ws^2			
C	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss
1	9839	9.96	157	2473	2.23	76	55	1857	2.31	53	38
4	9087	11.25	140	2140	2.33	68	54	2487	2.36	66	44
8	11613	8.47	143	2338	2.45	66	45	1813	2.30	52	30
16	38617	10.26	242	3403	2.66	77	39	2570	2.26	58	23
1	22246	14.90	165	4811	3.31	87	76	4668	3.06	66	55
4	17976	18.22	131	4324	2.57	77	64	4373	3.19	66	45
8	30460	18.18	159	5224	3.14	85	60	4209	2.86	57	36
16	74447	16.50	176	5532	3.14	67	46	5191	3.02	64	23
1	23771	11.89	149	9215	2.96	97	78	6815	3.01	69	56
4	28567	10.97	176	6766	2.99	79	63	6506	3.07	69	45
8	27871	12.07	130	7560	2.67	87	56	5765	2.78	61	37
16	58363	13.95	156	8626	3.14	83	45	3764	2.95	41	18

- SDW always slower, ws^2 most often faster, S^2DW gaps much better

Conclusions

Conclusions and (a lot of) future work

- Multicommodity flows is a veritable mine of structures
- DW decomposition is a very old idea, very 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 (at least $\|\cdot\|_2^2$)
  - Need to find modern equivalent of [23] to exploit the structure of an unstructured problem (perhaps less contradictory than is sounds [32])
- Huge challenge: make these techniques mainstream
- A new hope: automatic reformulation techniques [33]

[32] Kiwiel "An alternating linearization bundle method for . . . and nonlinear multicommodity flow problems" *Math. Prog.* 2013
[33] F., Perez Sanchez "Transforming mathematical models using declarative reformulation rules" *LNCS* 6683, 2011

Exercises

Exercises

1. Prove (8)–(9) by LP duality
2. Prove QP duality and LP duality out of Lagrangian duality, then write the "mixed" case where some variables have all-0 Q coefficients
3. Give a graphical representation of $z \in \partial_\varepsilon f(x)$
4. Prove properties i)–iv) of f^* (not necessarily in this order)
5. Compute the Fenchel's conjugate of the following functions:
   $f(x) = cx$, $f(x) = x^T Qx/2$ for $Q \succ 0$,
   $f(x) = \max\{ a_i x + b_i \;\; i = 1, \ldots, p \}$, $f(x) + \alpha$, $f(x + \beta)$,
   $f(\alpha x)$ for $\alpha \neq 0$, $\alpha f(x)$ for $\alpha > 0$, $I_X(x) = 0$ if $x \in X$, $+\infty$ otherwise
6. "Prove" Fenchel's duality: where does the odd "$-$" comes from?
7. Prove QP duality and LP duality out of Fenchel's duality
8. Prove (10)–(11)
9. Prove (12): who is f^* for our dual f?

Exercises (cont.d)

10. Prove (13)
11. Homage to Fenchel: prove (14) and (13) with $\mathcal{D} = \frac{1}{2t}\|\cdot\|_2^2$ by QP duality (tedious!!)
12. Plot (15) and (16), then compute their conjugate (and that of (17) if you dare)
13. Prove the two formulæ at Slide 46, then extend the second to the nonlinear (convex) case $G(x_2) \leq g$ (warning: nontrivial, and assumptions are required)
14. Compute the conjugate of the Kleinrock delay function
15. "Prove" (21) computing the dual of (20) (choose your dual)
16. Prove (22)–(23) (no choice now)
17. Final effort! Prove (24), then extend to the nonlinear (convex) case $G(x_2) \leq g$ (see exercise 13)