Decomposition methods for large-scale programming III: advanced (not-so-old) theory

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

ÖGOR Workshop for PhD students 2022
Krems an der Donau — August 2, 2022

Meta–Outline

- Part I: basic (old) theory
- Part II: some (new) tools
- Part III: advanced (not-so-old) theory
- Part IV: a challenge to you

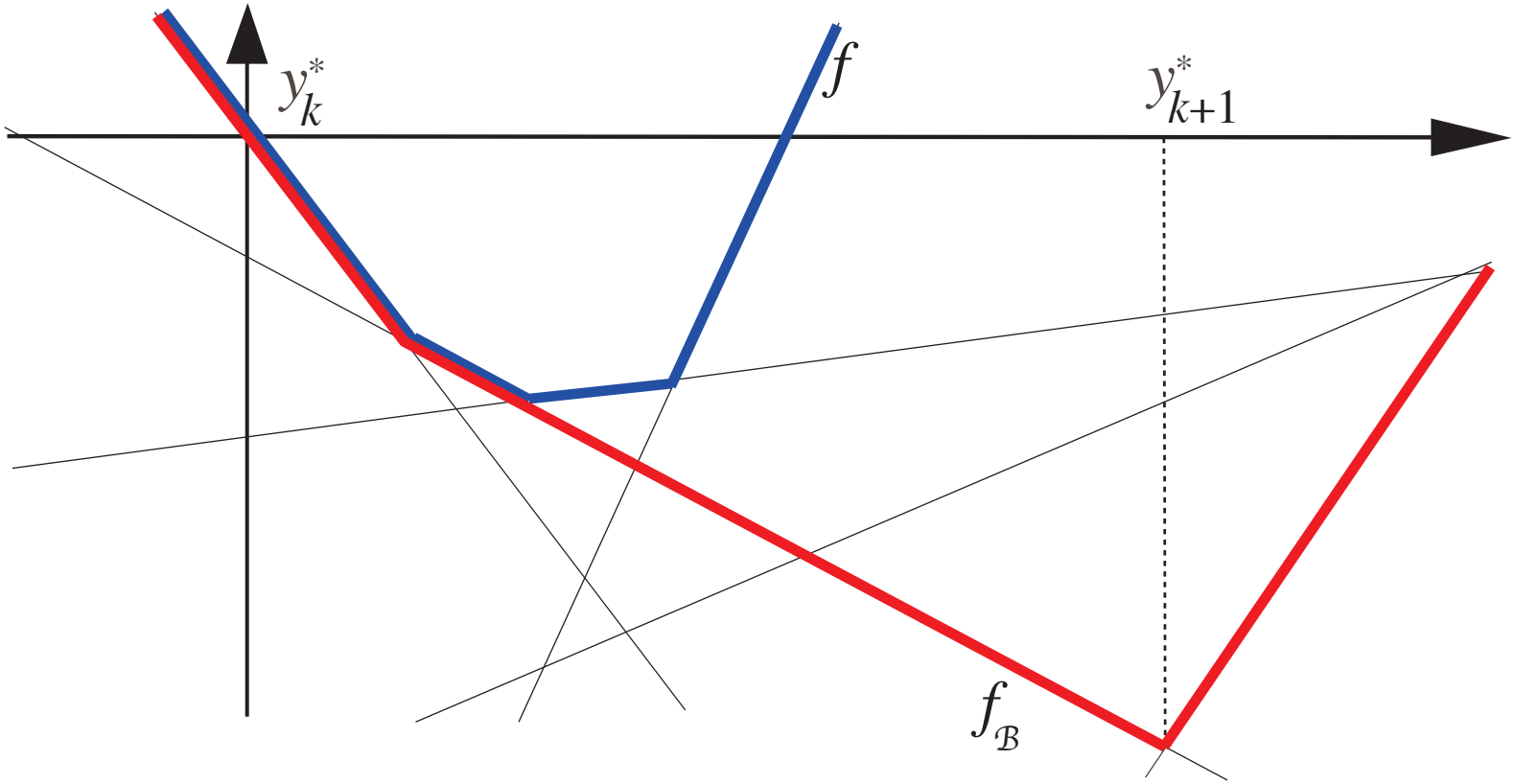
Outline – Part III

1. Stabilization
2. Disaggregated Model
3. Dual-Optimal Cuts
4. Cuts Selection
5. Easy Components
6. Structured Decomposition
7. Incremental, Inexact, Asynchronous
8. Conclusions (for now)

Stabilization

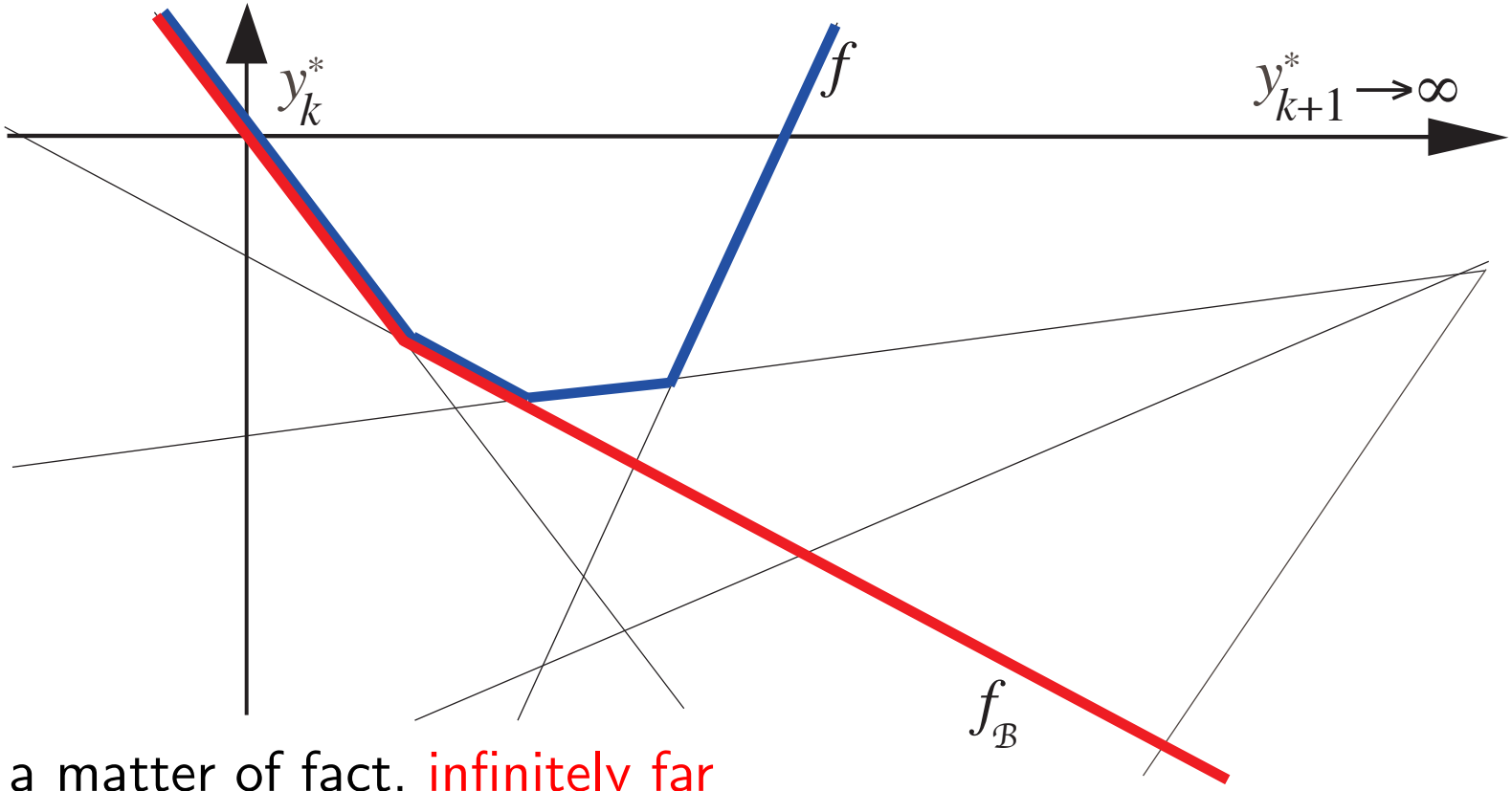
Issue with the Cutting-Plane approach: instability

- 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

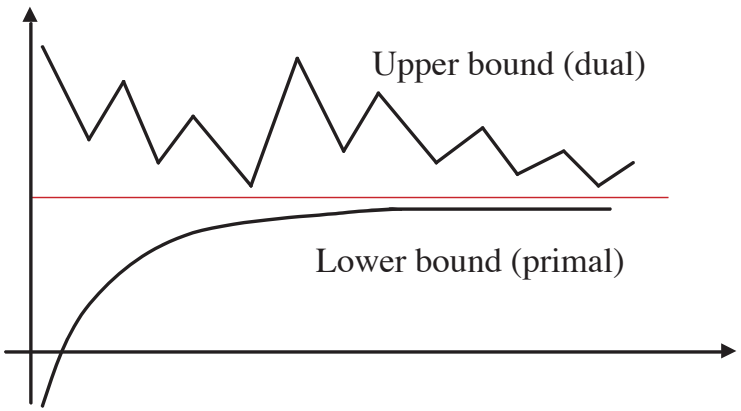
- $(\Pi_{\mathcal{B}})$ empty $\equiv$ $(\Delta_{\mathcal{B}})$ unbounded $\Rightarrow$ Phase 0 / Phase 1 approach

- 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

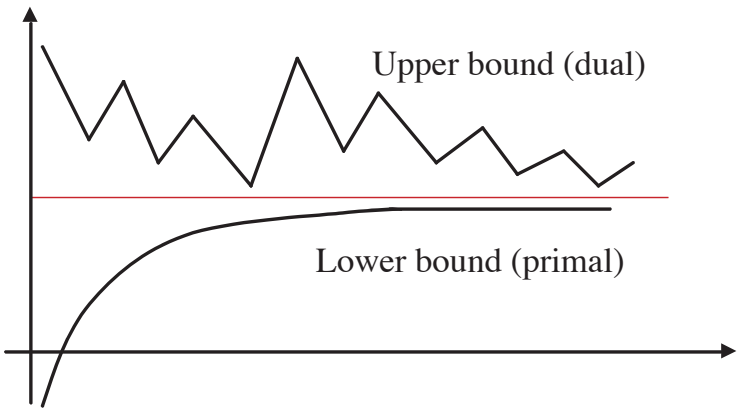
$\Longrightarrow$ tailing off, slow convergence



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)

Stabilizing DW/Lagrange/CG

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

- "If you don't know it guess it, but be ready to revise your guess"

Stabilizing DW/Lagrange/CG

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

- "If you don't know it guess it, but be ready to revise your guess"
- Current point $\bar{y}$, box of size $t > 0$ (how chosen??) around it
- Stabilized dual master problem[1] (variable = displacement from $\bar{y}$)

$$(\Delta_{\mathcal{B},\bar{y},t}) \qquad \min \left\{ f_{\mathcal{B}}(\bar{y} + d) \ : \ \| d \|_\infty \leq t \right\} \tag{1}$$

- Corresponding stabilized primal master problem

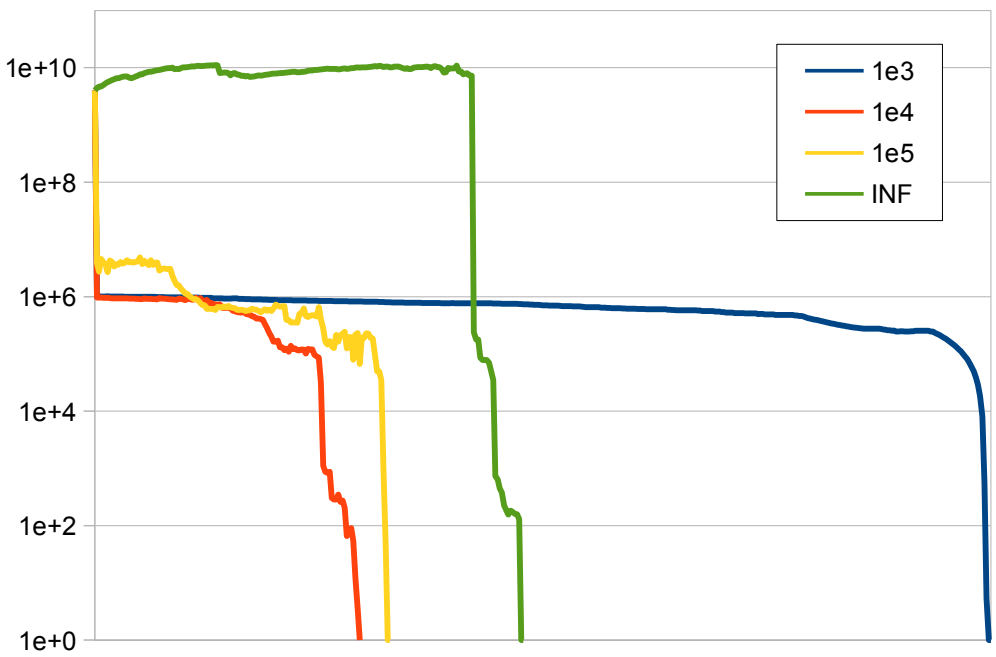
$$(\Pi_{\mathcal{B},\bar{y},t}) \qquad \max \left\{ cx + \bar{y}s - t\| s \|_1 \ : \ s = b - Ax \ , \ x \in conv(\mathcal{B}) \right\}$$

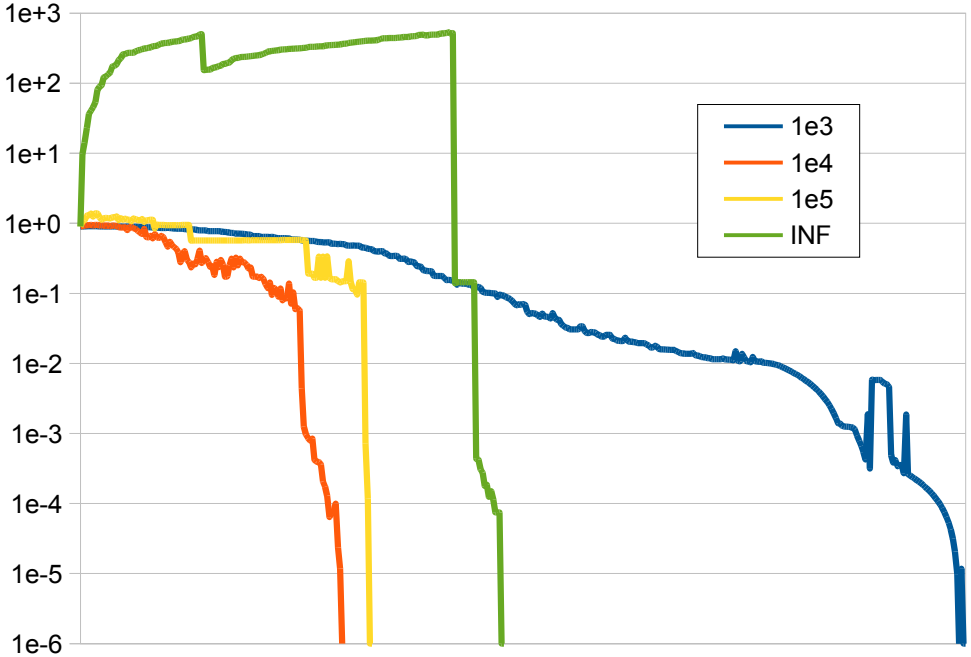
  i.e., just Dantzig-Wolfe with slacks (s)

- When $f(\bar{y} + d^*) \ll f(\bar{y})$, move $\bar{y} = \bar{y} + d^*$ ("serious step")
- Uses just LP tools, relatively minor modifications to $(\Delta_{\mathcal{B}})$
- Does this really work?

[1] Marsten, Hogan, Blankenship "The Boxstep Method for Large-scale Optimization" *Op. Res.*, 1975

Computational results of the boxstep method (pds7)

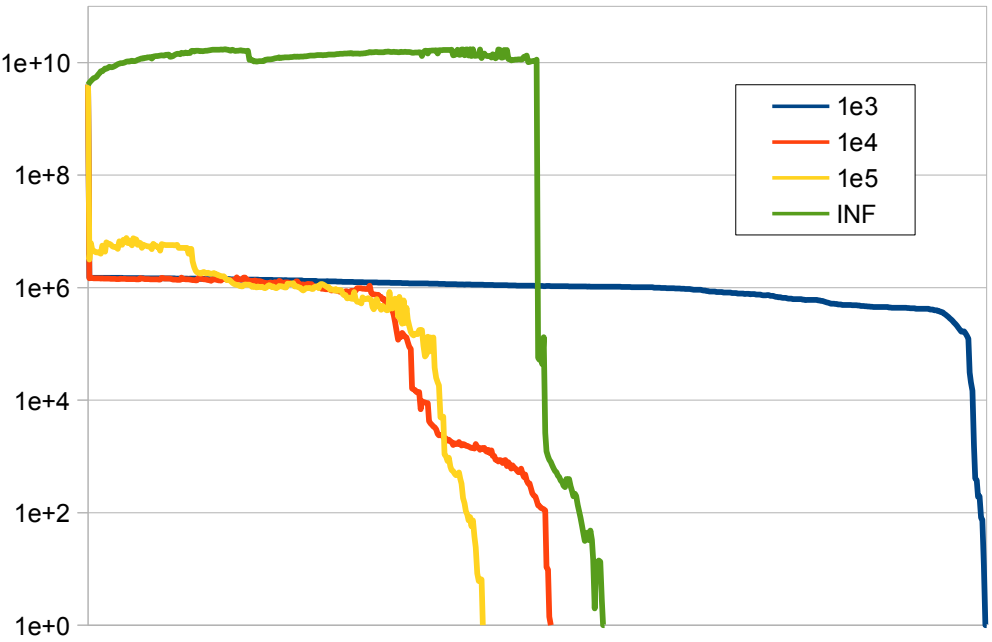


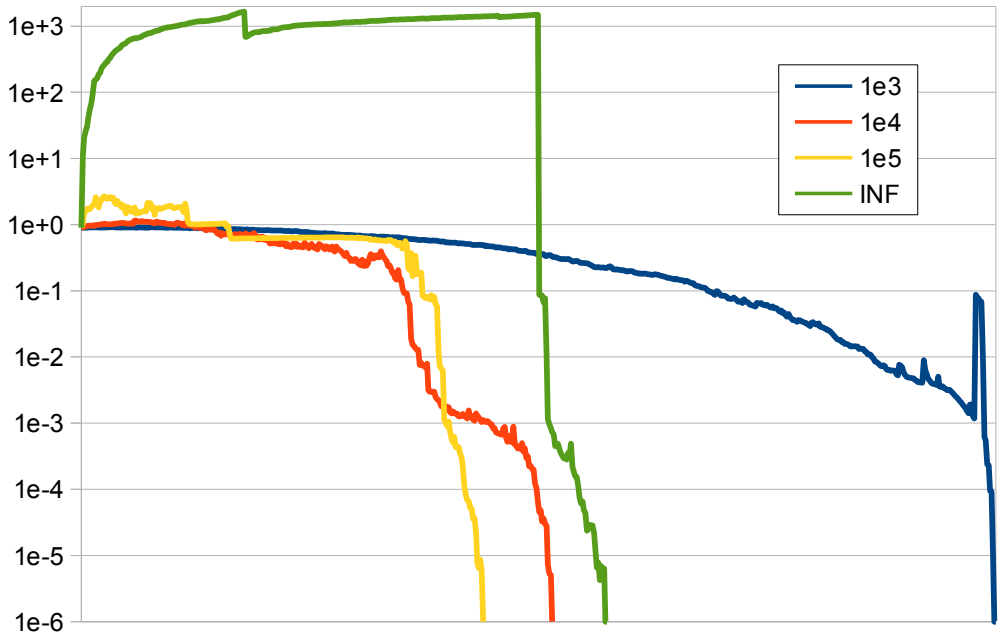


- Left = distance from final dual optimum

  right = relative gap with optimal value
- Stabilized with (fixed) different t, un-stabilized ($t = \infty$)
- One can clearly over-stabilize
- But non-stabilised "crazy", iterates' quality degrades by 3+ orders of magnitude and stays there for a long time until some "magic" kicks in

Computational results of the boxstep method (pds18)

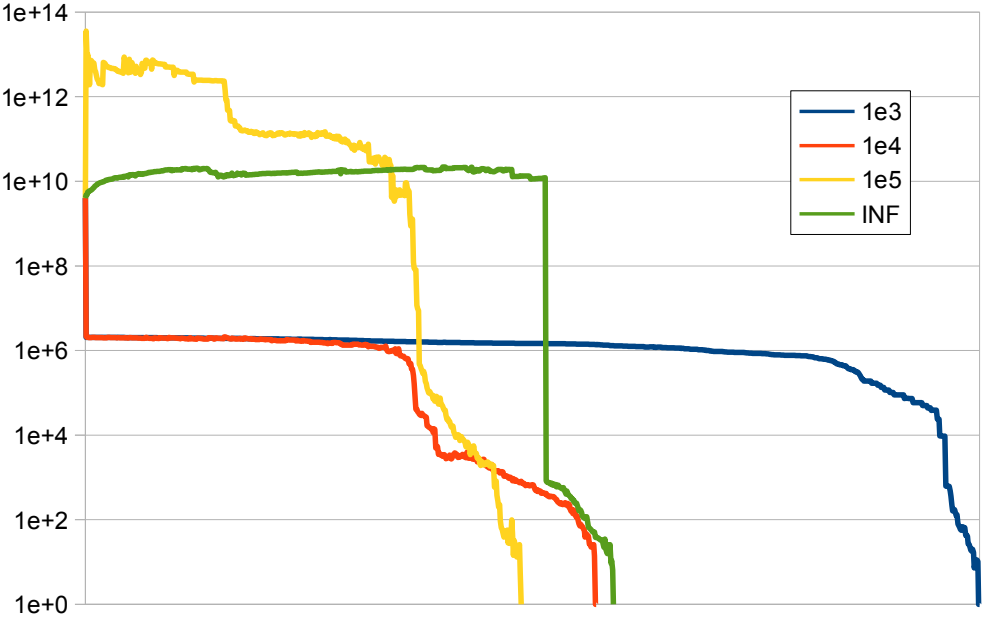


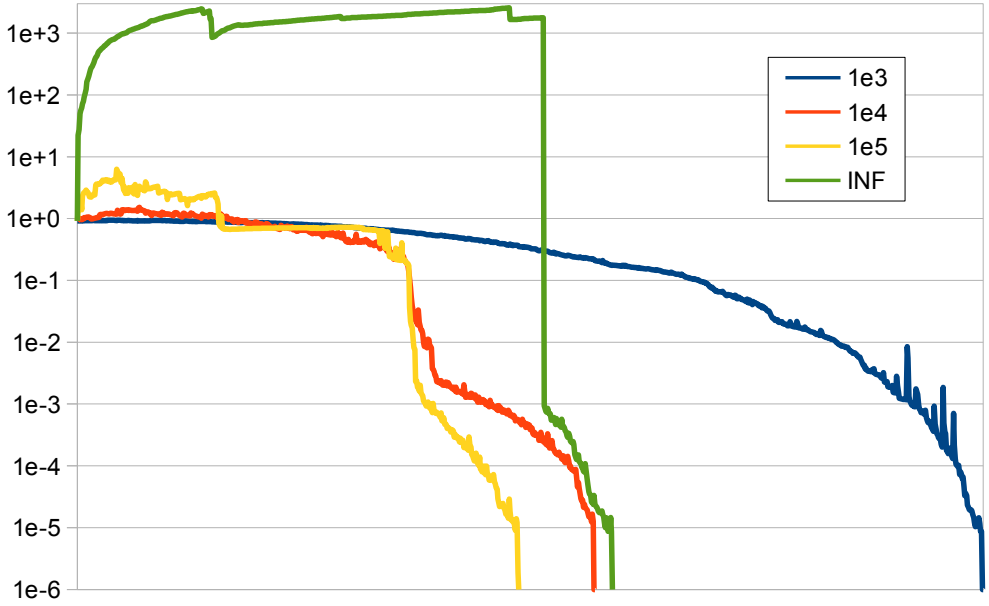


- All cases show a "combinatorial tail" where convergence is very quick
- $t =$ 1e+3: "smooth but slow" until the combinatorial tail kicks in, a short-step approach not unlike subgradient methods[2]
- $t = \infty$: apparently trashing along until some magic threshold is hit
- "intermediate" t work best, but pattern not clear

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

Computational results of the boxstep method (pds30)





- $t =$ 1e+5: 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[3]

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

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

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

Generalized proximal/trust region stabilization

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

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

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

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

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

  where $f^*(\,x\,) = \max_s\{\, xs - f(\,s\,)\,\}$ the Fenchel's conjugate of f

- For our dual f, a generalized augmented Lagrangian

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

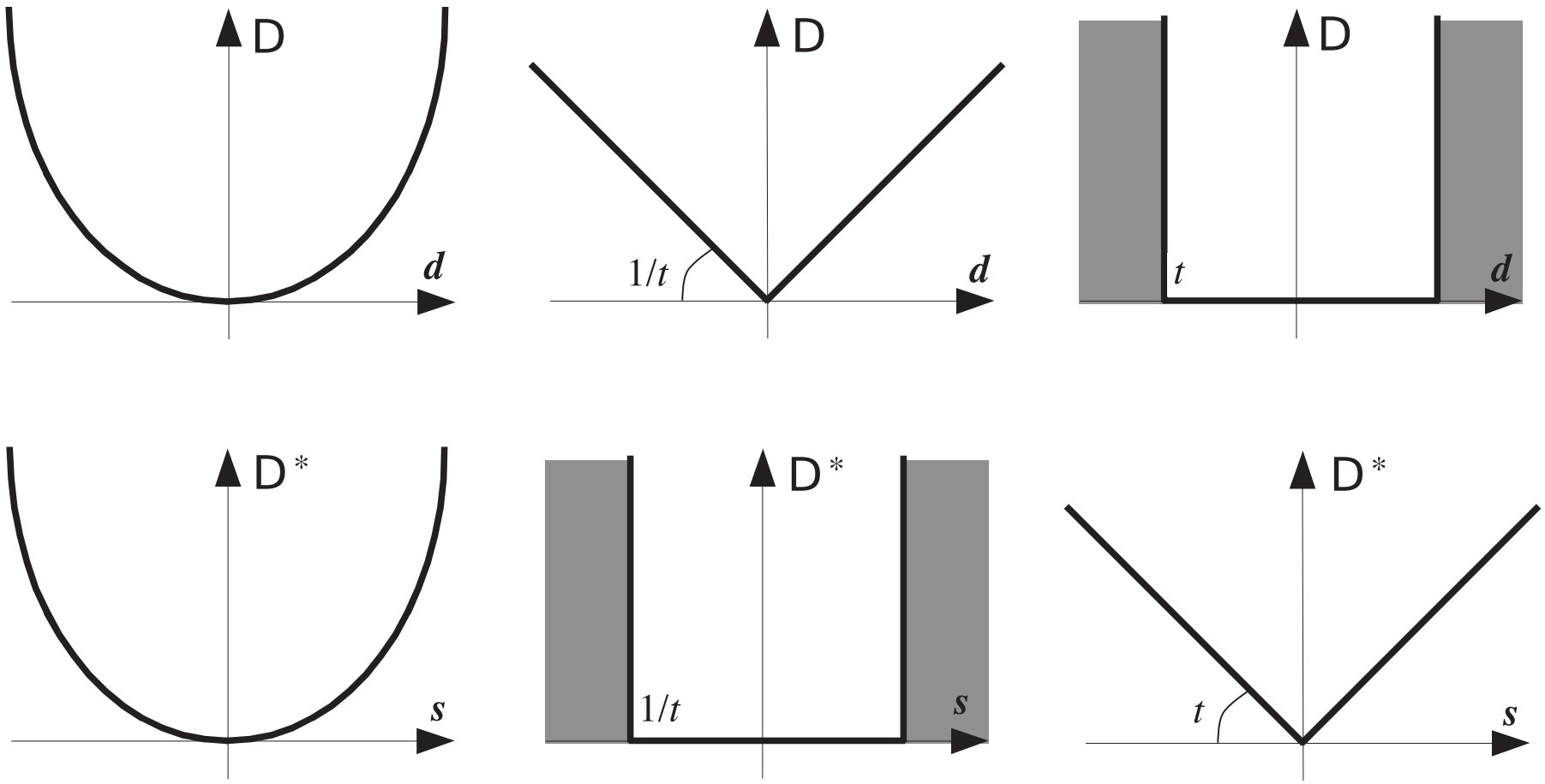
- A "primal" exists even for a non-dual f: $v(\Pi) = -f^*(0) = v(\Delta)$ for

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

- General theory exist[4], but never mind

[4] F. "Generalized Bundle Methods" *SIOPT*, 2002

Classical stabilizing terms



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

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

Fancier stabilizing terms (very nonlinear)

- Smooth approximation of $\|\cdot\|_1$[5]

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

- Smooth approximation of $t\|\cdot\|_\infty$[6]

$$\mathcal{D}^*(\,s\,) = \ln \sum_i e^{ts_i}$$

- Bregman functions[7]

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

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

- Others (φ-divergences, ...), all "very nonlinear"

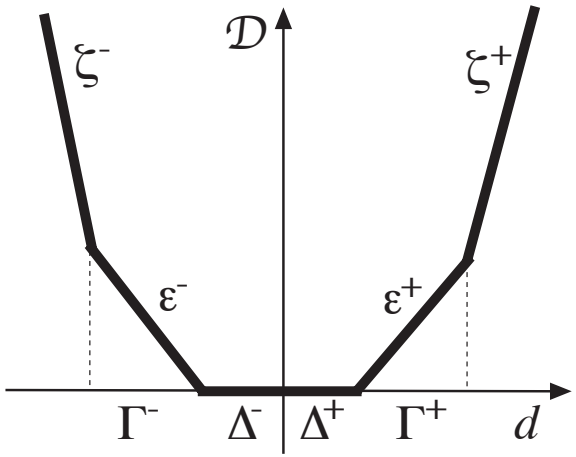
[5] Pinar, Zenios "Parallel Decomposition of Multicommodity [...] Using a Linear-Quadratic [...]" *ORSA J. Comp.*, 1992

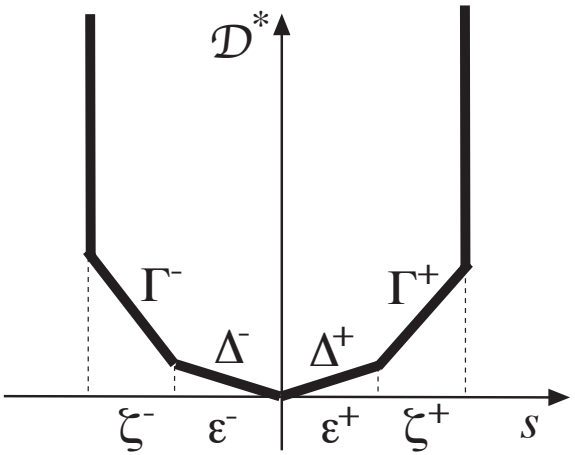
[6] Grigoriadis, Khachiyan "An Exponential Function Reduction Method for Block-Angular Convex Programs" *Networks*, 1995

[7] Chen, Teboulle "Convergence Analysis of a Proximal-like Minimization Algorithm Using Bregman Functions" *SIOPT*, 1993

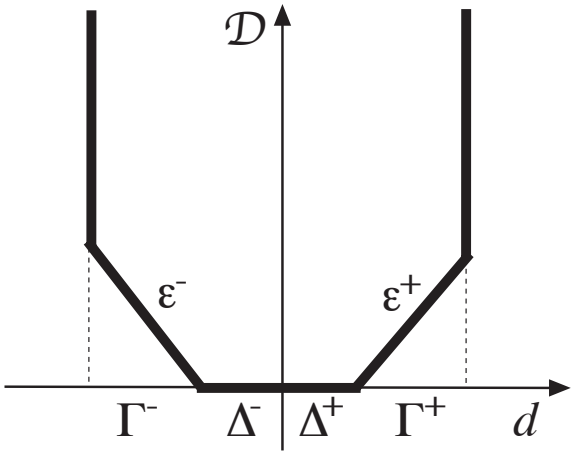
A 5-piecewise-linear function

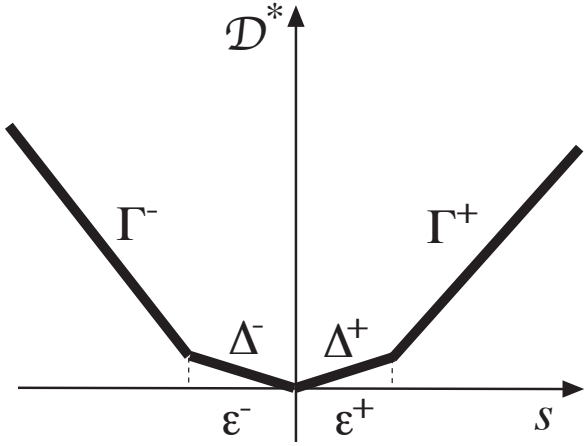
Trust region on $\bar{y}$ + small penalty close + much larger penalty farther[8]





Slightly simplified version: only 3 pieces.





[8] Ben Amor, Desrosiers, F. "On the Choice of Explicit Stabilizing Terms in Column Generation" *DAM*, 2009

A 5-piecewise-linear master problem

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

A 5-piecewise-linear master problem

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

- 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 even 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

- Recall: ray χ of X: $x \in X \Longrightarrow x + \lambda\chi \in X$ for $\lambda \to \infty$
- $(c - yA)\chi > 0 \Longrightarrow f(\,y\,) = \infty \Longrightarrow$ constraint $c\chi \leq y(A\chi)$ in the dual

On unboundedness and early termination

- Recall: ray χ of X: $x \in X \Longrightarrow x + \lambda\chi \in X$ for $\lambda \to \infty$
- $(c - yA)\chi > 0 \Longrightarrow f(\,y\,) = \infty \Longrightarrow$ constraint $c\chi \leq y(A\chi)$ in the dual
- One might even hide the convexity constraint:
  - $A\bar{x} \to [\,A\bar{x}\,,\,1\,] \quad , \quad b \to [\,b\,,\,1\,]$;
  - Ignoring the special role of v (just another y)
  - Advantage: everything is a constraint, standard in CG

On unboundedness and early termination

- Recall: ray χ of X: $x \in X \Longrightarrow x + \lambda\chi \in X$ for $\lambda \to \infty$
- $(c - yA)\chi > 0 \Longrightarrow f(\,y\,) = \infty \Longrightarrow$ constraint $c\chi \leq y(A\chi)$ in the dual
- One might even hide the convexity constraint:
  - $A\bar{x} \to [\,A\bar{x}\,,\,1\,] \quad , \quad b \to [\,b\,,\,1\,]$;
  - Ignoring the special role of v (just another y)
  - Advantage: everything is a constraint, standard in CG

$\boxed{\text{This is a bad idea!}}$

On unboundedness and early termination

- Recall: ray χ of X: $x \in X \Longrightarrow x + \lambda\chi \in X$ for $\lambda \to \infty$
- $(c - yA)\chi > 0 \Longrightarrow f(\,y\,) = \infty \Longrightarrow$ constraint $c\chi \leq y(A\chi)$ in the dual
- One might even hide the convexity constraint:
  - $A\bar{x} \to [\,A\bar{x}\,,\,1\,] \quad , \quad b \to [\,b\,,\,1\,]$;
  - Ignoring the special role of v (just another y)
  - Advantage: everything is a constraint, standard in CG

$\boxed{\text{This is a bad idea!}}$

- Moving $\bar{y}$ requires testing for decrease in f-value, but when a ray is generated, $f(\,\bar{y} + d^*\,) = \infty$
- Ignoring convexity constraint $\Longrightarrow$ Proximal Point: solve the problem exactly for $\bar{y}$ before moving it
- Convexity constraints are good: invent them if they are not there
- In fact, the more convexity constraints the better (will see soon)

Computational Results: the effect of early termination

- Same setting as before
- Comparing 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
- No reason to solve with high accuracy if $\bar{y}$ not (close to) optimal
- General concept, we'll see it again

Other Forms of Stabilization

- Proximal level[9]: closest point promising given amount of decrease

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

- l somehow easier to manage than t, easy rules available that allow to keep $\bar{y}$ fixed (but possible in proximal, too)
- Trade blows in practice, but doubly-stabilised possible[10]
- Different approach: aim for center (analytic[11] or Chebychev[12]) of localization set $\mathcal{L} = \{ (y, v) : f_{\mathcal{B}}(y) \leq v \leq f(\bar{y}) \} \subset \mathbb{R}^{n+1}$
- "Good" theoretical performances, but in practice a penalty term may still be required[13]

---

[9] Lemaréchal, Nemirovskii, Nesterov "New Variants of Bundle Methods" *Math. Prog.*, 1995

[10] de Oliveira, Solodov "A Doubly-Stabilized Bundle Method for Nonsmooth Convex Optimization" *Math. Prog.*, 2016

[11] Gondzio, González-Brevis, Munari "New Developments in the Primal-Dual Column Generation Technique" *EJOR*, 2013

[12] Ouorou "A proximal cutting plane method using Chebychev center for nonsmooth convex optimization" *Math. Prog.*, 2009

[13] Babonneau, Beltran, Haurie, Tadonki, Vial "Proximal-ACCPM: a Versatile Oracle-Based [...]" *Adv. Comp. Man. Sci.* 2007

From Minimally to Maximally Intrusive Stabilization

- Changing the master problem not strictly needed: In-Out approach[14] computes un-stabilised d^* (if $\exists$) but probes $f(\bar{y} + \alpha d^*)$, $\alpha \in (0, 1]$
- Simple to implement and can still work well in practice[15]
- Other extreme: $\mathcal{D}(d) = d^T Q d$, $Q =$ "approximation of $\nabla^2 f(\bar{y})$" a-la quasi-Newton (Hessian of a piecewise-linear function ?!?!)
- Theory exists, superlinear convergence possible[16]
- Hard to make work in practice, but simpler scalings may work already[17]
- Many nice ideas[18] if you like the research line, work in practice
- But parameters $(t, l, \alpha, \dots)$ tuning still an art more than a science

---

[14] Ben-Ameur, Neto "Acceleration of Cutting-Plane and Column Generation Algorithms [...]" *Networks*, 2007

[15] Pessoa, Sadykov, Uchoa, Vanderbeck "Automation [...] of [...] Stabilization [...] in Column Generation" *IJoC*, 2018

[16] Mifflin, Sagastizábal "A $\mathcal{VU}$-algorithm for Convex Minimization" *Math. Prog.*, 2005.

[17] Helmberg, Pichler "Dynamic Scaling and Submodel Selection in Bundle Methods for Convex Optimization" *OO*, 2017

[18] F. "Standard Bundle Methods: Untrusted Models and Duality" *Numerical Nonsmooth Optimization*, 2020

Stabilized Benders' Decomposition

- Easy with trust region, pretty identical to (1) (but no dual)

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

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

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

- However, $x^* = \bar{x}$ only $\Longrightarrow$ $\bar{x}$ local optimum (nonconvex)
  $\Longrightarrow$ have to increase t until $t = n$ (∞)
- Silver lining: reverse box $\| \, x - \bar{x} \, \|_\infty \geq t$ (nonconvex) now easy
- Level stabilization a-la (2) also possible[19], pros and cons:
  $(B_{\mathcal{B},\bar{x},l})$ can be solved inexactly (but larger and more difficult),
  l easier to manage than t and need not go ∞ (but no reverse box)
- All in all it does work[20] (but nontrivial)

---

[19] van Ackooij, F., de Oliveira "Inexact Stabilized Benders' Decomposition Approaches [...]" *COAP*, 2016

[20] Baena, Castro, F. "Stabilized Benders Methods for Large-scale Combinatorial Optimization [...]" *Man. Sci.*, 2020

Computational Results: Cell Suppression Problem

classic				stabilized			
iter	gap	Tot	TM	iter	gap	Tot	TM
3	0.0	23	0	35	0.0	246	7
3	0.0	91	1	49	0.0	1344	6
2	0.0	22	0	39	0.0	427	3
6	0.0	310	1	37	0.0	1851	4
133	18.7	***	1920	259	0.0	***	157
29	18.8	***	3438	472	1.0	***	749
237	14.1	***	1542	321	0.5	***	383
223	7.5	***	956	269	0.3	***	162
225	16.6	***	2917	672	0.0	***	1193
43	7.3	***	2681	168	0.1	***	117
218	9.3	***	342	219	1.0	***	116
14	16.0	***	3294	151	2.4	***	103
27	20.7	***	3201	232	2.8	***	39
147	18.7	***	190	131	1.2	***	51

- Combinatorial problem (cell suppression), time limit 3600s
- Don't stabilize easy stuff, but do stabilize difficult one
- More iterations, but master problem time per iteration greatly reduced

Disaggregated Model

Disaggregated Model for the Block-diagonal Program

- The real decomposition case:

$$(\Pi) \quad \max \left\{ \sum_{k \in K} c^k x^k \ : \ \sum_{k \in K} A^k x^k = b \ , \quad x^k \in X^k \quad k \in K \right\}$$

i.e., $\bar{x} = [\,\bar{x}^k\,]_{k \in K}$ (Cartesian product of individual solutions)

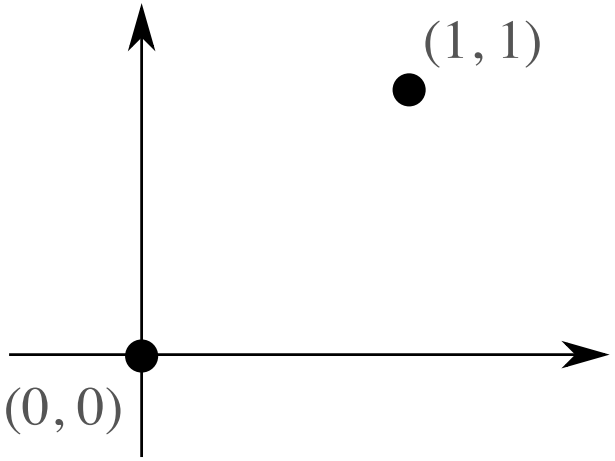
- Disaggregated DW reformulation:

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

i.e., $X = X^1 \times X^2 \times \ldots \times X^{|K|} \Longrightarrow$
$conv(X) = conv(X^1) \times conv(X^2) \times \ldots \times conv(X^{|K|})$

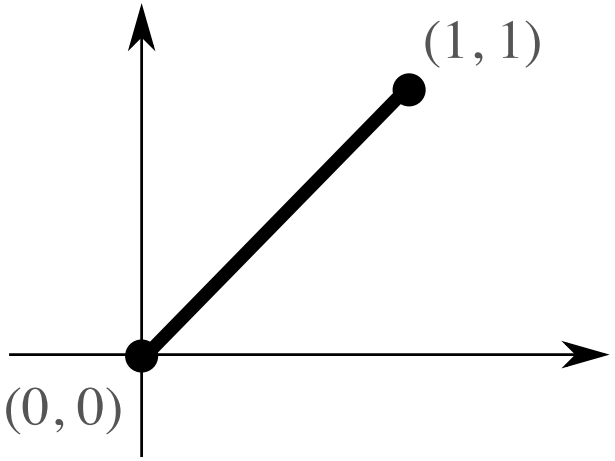
- A different multiplier $\theta^k_{\bar{x}}$ for each $k \in K$: aggregated is $\theta^k_{\bar{x}} = \theta^h_{\bar{x}}$ for $h \neq k \Longrightarrow$ a restriction (less solutions $\equiv$ bad)

Geometry of Disaggregated Models



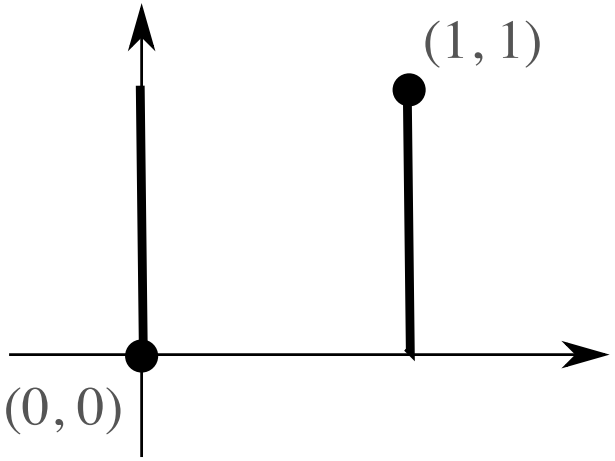
- Given X,

Geometry of Disaggregated Models



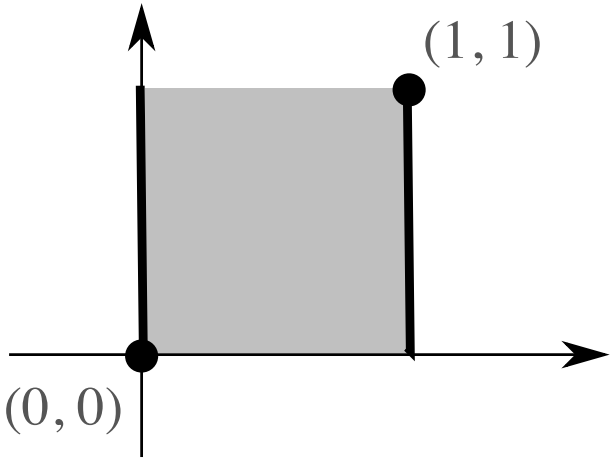
- Given X, taking the convex hull of Cartesian products

Geometry of Disaggregated Models



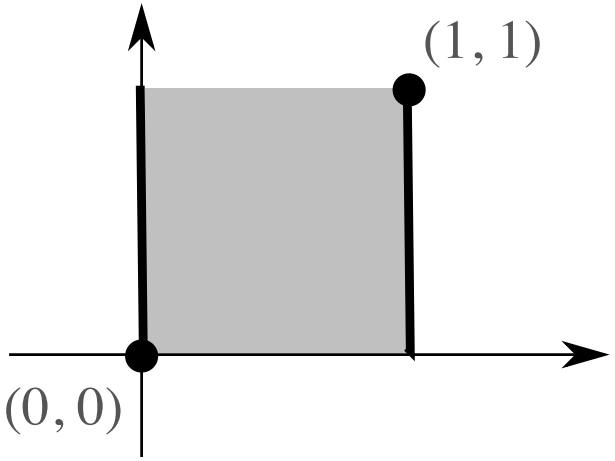
- Given X, taking the convex hull of Cartesian products is smaller (bad) than first making convex hulls

Geometry of Disaggregated Models



- Given X, taking the convex hull of Cartesian products is smaller (bad) than first making convex hulls and then taking the Cartesian product

Geometry of Disaggregated Models



- Given X, taking the convex hull of Cartesian products is smaller (bad) than first making convex hulls and then taking the Cartesian product

- From the dual viewpoint

$$f_{\mathcal{B}}(\, y\,) = \sum_{k \in K} f^k_{\mathcal{B}}(\, y\,)$$

the sum of individual models is better than the model of the sum

Disaggregated Dantzig-Wolfe and Multicommodity flows

- Aggregated DW: $\mathcal{S} = \{$ (extreme) flows $s = [\bar{x}^{1,s}, \ldots, \bar{x}^{k,s}] \}$

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

- Disaggregated + scaling $\equiv$ path-flow formulation:

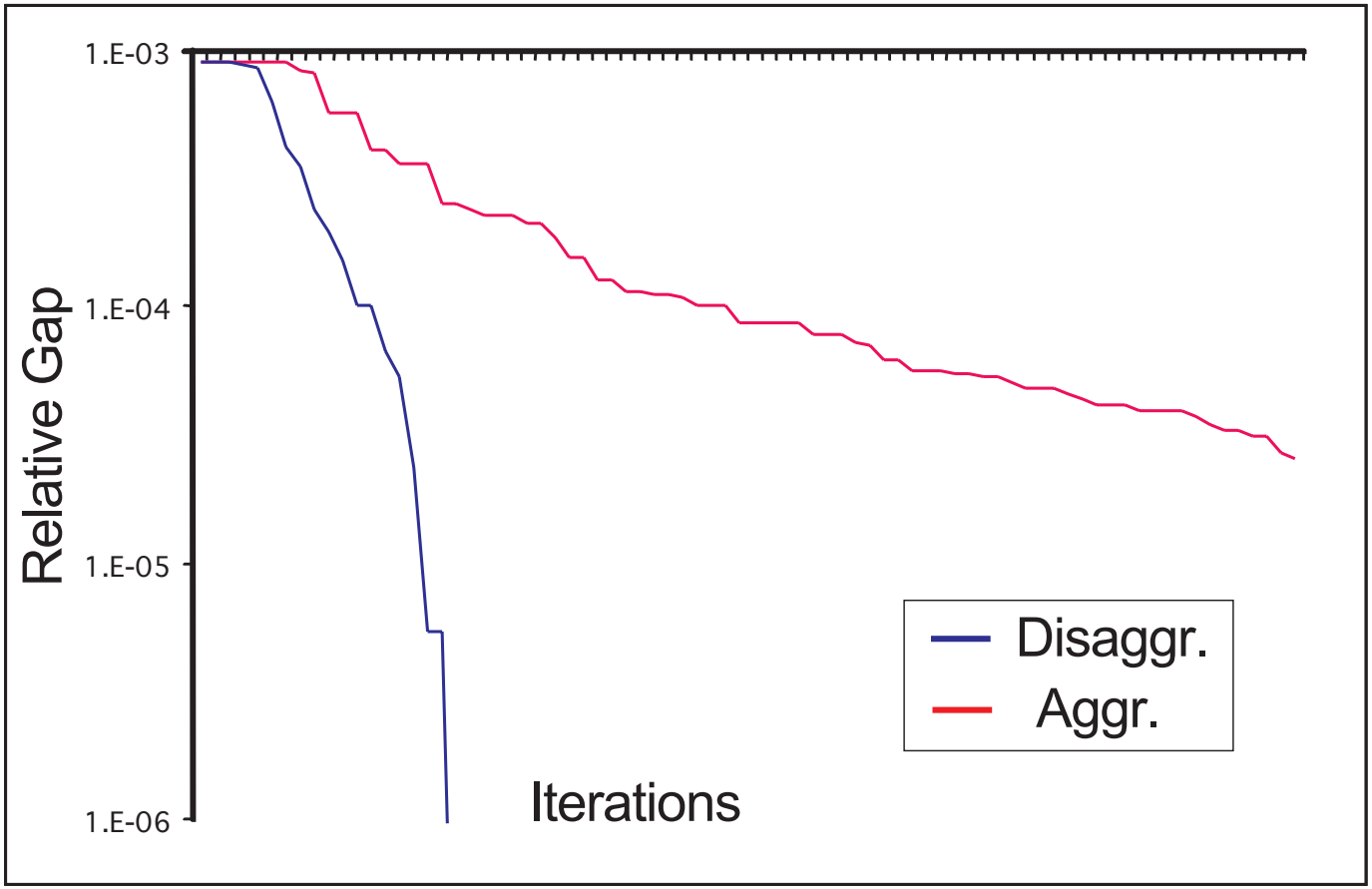
$p \in \mathcal{P}^k = \{ s^k\text{–}t^k$ paths $\}$, c_p cost, $f_p (= d^k \theta_s^k)$ flow, $\mathcal{P} = \cup_{k\in K} \mathcal{P}^k$

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

- More columns but sparser, (a few) more rows, much more efficient[21]
- One simplex constraints vs. many: the more, the merrier

[21] Jones, Lustig, Farwolden, Powell "Multicommodity Flows: the Impact of Formulation on Decomposition" *Math. Prog.* 1993

Disaggregated decomposition: computational impact



- Easily extended to any decomposable X[22], stabilized versions immediate
- Much fewer iterations, but larger master problems $\equiv$ trade-offs

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

More or Less Disaggregated?

- That was ≈ 30 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, subproblems remain the same
- How to choose the C_i? In general open problem
- Aggregation can be dynamic[23], even more open problem, but it can work[17]

[23] van Ackooij, F. "Incremental Bundle Methods Using Upper Models" *SIOPT*, 2018

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 not to (can always recover the original solution, less variables)
- This impact how the master problem is formulated[21] ...

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 not to (can always recover the original solution, less variables)
- This impact how the master problem is formulated[21] ... 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[21]
- Again, commodities in a C_i just have "close destination node names"

Dealing With Multiple-origin Commodities

- What about commodities that have many origins (PSP in [21])?
- Can always assume one origin (add a super-origin) ...

Dealing With Multiple-origin Commodities

- What about commodities that have many origins (PSP in [21])?
- Can always assume one origin (add a super-origin) ...
  but must add commodity-specific capacities on super-origin arcs
- Data no longer commodity-independent, subproblems no longer SPTs
  $\Longrightarrow$ cannot disaggregate by origin ...

Dealing With Multiple-origin Commodities

- What about commodities that have many origins (PSP in [21])?
- Can always assume one origin (add a super-origin) . . .
  but must add commodity-specific capacities on super-origin arcs
- Data no longer commodity-independent, subproblems no longer SPTs
  $\Longrightarrow$ cannot disaggregate by origin . . . or you can:
  "just" consider individual capacities as complicating constraints
- This is still a reformulation of the master problem,
  has (almost) nothing to do with the original problem formulation
- Obvious trade-off: simpler subproblems, harder master problem
  (possibly many more rows, more columns but sparser ones)
- TTBoOK haven't been computationally explored to far

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

- Generalized Bundle code using $D_t^* = \| \cdot \|_1$ (boxstep)
- Latest Cplex as Master Problem Solver
- Implementation issue: overhead due to subgradient handling significant
- Limited effect of stabilization (not much need)
- (Reasonably) efficient subproblem solution with `MCFClass`
  https://github.com/frangio68/Min-Cost-Flow-Class
- Many instances, some old, some new, from
  https://commalab.di.unipi.it/datasets/mmcf
- Results for $\eta < 0$ still brewing, but these significant enough already
- In a nutshell: there is great chaos under heaven

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
- *** = 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 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 $\eta = 1$)
- Strange(r) things happen ($\eta = 1$ can even be best)

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[24]
- Drawback: not general approach, cuts are problem-specific

[24] 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 (path-flow formulation)

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

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 (path-flow formulation)

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

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
- Some good results, other applications (Cutting Stock, different cuts)

Cuts Selection

(Feasibility) Cuts Selection

- $v(\,x\,) = -\infty \Longrightarrow$ any $\bar{\omega} \in W_\infty$ gives a cut: which one is "best"?
- If LP solver choses, can't expect it to pick a "good one"
- $(\,x^*\,,\,v^*\,)$ solution of $(B_{\mathcal{B}})$: a cut does not $\exists \iff$

$$v(\,x^*\,) = \max\{\; ez \;:\; Ez \leq d - Dx \;\} \geq v^*$$

$$\equiv \;\; \max\{\; 0z \;:\; ez \geq v^* \;,\; Ez \leq d - Dx^* \;\} = 0$$

- Hence a cut does $\exists \iff$

$$\min\{\; w(d - Dx^*) - w_0 v^* \;:\; wE = w_0 e \;,\; (\,w\,,\,w_0\,) \geq 0 \,\} = -\infty$$

$\equiv$ (homogeneity)

$$0 > \min\; w(d - Dx^*) - w_0 v^*$$

$$wE = w_0 e \;,\; w\beta + w_0\beta_0 = 1 \;,\; (\,w\,,\,w_0\,) \geq 0$$

however chosen $(\,\beta\,,\,\beta_0\,)$: a proper choice improves performances[25]

[25] Fischetti, Salvagnin, Zanette "A Note on the Selection of Benders' Cuts" *Math. Prog.*, 2010

Easy Components

Decomposition of Multicommodity Network Design

- Multicommodity flow + arc design costs f_{ij}, design variables $z_{ij} \in \{0, 1\}$
- D-W reformulation $\equiv$ path-flow for flows, what about design?

Decomposition of Multicommodity Network Design

- Multicommodity flow + arc design costs f_{ij}, design variables $z_{ij} \in \{0, 1\}$
- D-W reformulation $\equiv$ path-flow for flows, what about design?
- $\mathcal{S}$ = extreme points of z ($2^{|A|}$ vertices of the unitary hypercube):

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

Decomposition of Multicommodity Network Design

- Multicommodity flow + arc design costs f_{ij}, design variables $z_{ij} \in \{0, 1\}$
- D-W reformulation $\equiv$ path-flow for flows, what about design?
- $\mathcal{S}$ = extreme points of z ($2^{|A|}$ vertices of the unitary hypercube):

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

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

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

- ...and with very long, dense rows

Multicommodity Network Design, the Right Way

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

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

Multicommodity Network Design, the Right Way

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

  $\boxed{z_{ij} \in [\, 0 \, , \, 1 \,]}$ (no, ... really?!)
- Path-flow formulation with original arc design variables

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

- Only generate the right variables

Is it always this easy?

- No: what if one had, say,

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

- Design subproblem can no longer be disaggregated

Is it always this easy?

- No: what if one had, say,

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

- Design 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

Is it always this easy?

- No: what if one had, say,

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

- Design 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
- Partial D-W reformulation: do that on the right part only
- In Lagrangian parlance: some components are "easy"
- Can this be stabilized? Of course it can[26]

[26] F., Gorgone "Bundle Methods for Sum-Functions With "Easy" Components: [...] Network Design" *Math. Prog.*, 2014

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

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

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"

- 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 and implementable form

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

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

- Barring some details (do not translate $f^1_{\mathcal{B}}$), everything works
- Performances can improve dramatically (not hard to see why)

Extensions, details

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

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

- Known global lower bound $l \leq \min f$ (c_2 and G linear)

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

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

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[27])
- With (strong) or without (weak) forcing constraints

$$0 \leq x_{ij}^k \leq u_{ij}^k z_{ij} \qquad (i,j) \in A\,,\ k \in K$$

(very many, so dynamic generation[28,29] needed)

- https://commalab.di.unipi.it/datasets/mmcf/#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

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

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

[29] 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[30])

[30] 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 (or in a CAT scan)

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

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

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 – the final table

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 = subgradient $\approx$ FA-2 with very small $|\mathcal{B}|$
- Tuning not easy, a lot of pieces have to click
- Much faster than `Cplex` and anything else as $|A|$ and/or $|K|$ grows

The Easy Component Need Not Be Linear

- Nonlinear multicommodity routing:

$$\min \left\{ \sum_{(i,j)\in A} \frac{z_{ij}}{1-z_{ij}} \ : \ \langle \text{multicommodity flow} \rangle \ , \ z \in [0,1]^{|A|} \right\}$$

  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[31]
- Substantially improved performances

[31] Lemaréchal, Ororou, Petrou "A Bundle-type Algorithm for Routing in Telecommunication Data Networks" *COAP*, 2009

Structured Decomposition

Motivation: knapsack decomposition

- Relax the flow conservation constraints (from arc-flow formulation)

$$
\begin{array}{lll}
\min & \sum_{(i,j)\in A}\left(\sum_{k\in K}(d^k c_{ij}^k - \pi_i^k + \pi_j^k)x_{ij}^k + f_{ij}z_{ij}\right) & \\
& \sum_{k\in K} d^k x_{ij}^k \leq u_{ij} z_{ij} & (i,j)\in A \\
& x_{ij}^k \in [0,1] & (i,j)\in A,\ k\in K \\
& z_{ij} \in \mathbb{N} & (i,j)\in A
\end{array}
$$

- 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[32]

$$
\begin{array}{ccc}
a_k = d^k / u_{ij} & a(S) = \sum_{k\in S} a_k & S \subseteq K \\
\multicolumn{3}{c}{\sum_{k\in S} a_k(1 - x_{ij}^k) \geq (\, a(S) - \lfloor a(S) \rfloor\,)(\,\lceil a(S) \rceil - z\,)}
\end{array}
\tag{3}
$$

- $\bar{I}+$ = standard continuous relaxation + (3) $\equiv$ DW[33]

[32] Atamtürk "On Capacitated Network Design Cut-Set Polyhedra" *Math. Prog.* 2002

[33] 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			$I+$		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			$I+$		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: $z_{ij} \leq \lceil \sum_{k \in K} d^k / a_{ij} \rceil = T_{ij}$
- Pseudo-polinomially many segments $S_{ij} = \{ 1, \ldots, T_{ij} \}$ for z_{ij}

Reformulation III: Binary formulation B

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

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

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

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

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

... then original variables can be removed

- Up to now, continuous relaxation bound has not improved

Improved binary formulation $B+$

- Extended linking inequalities:

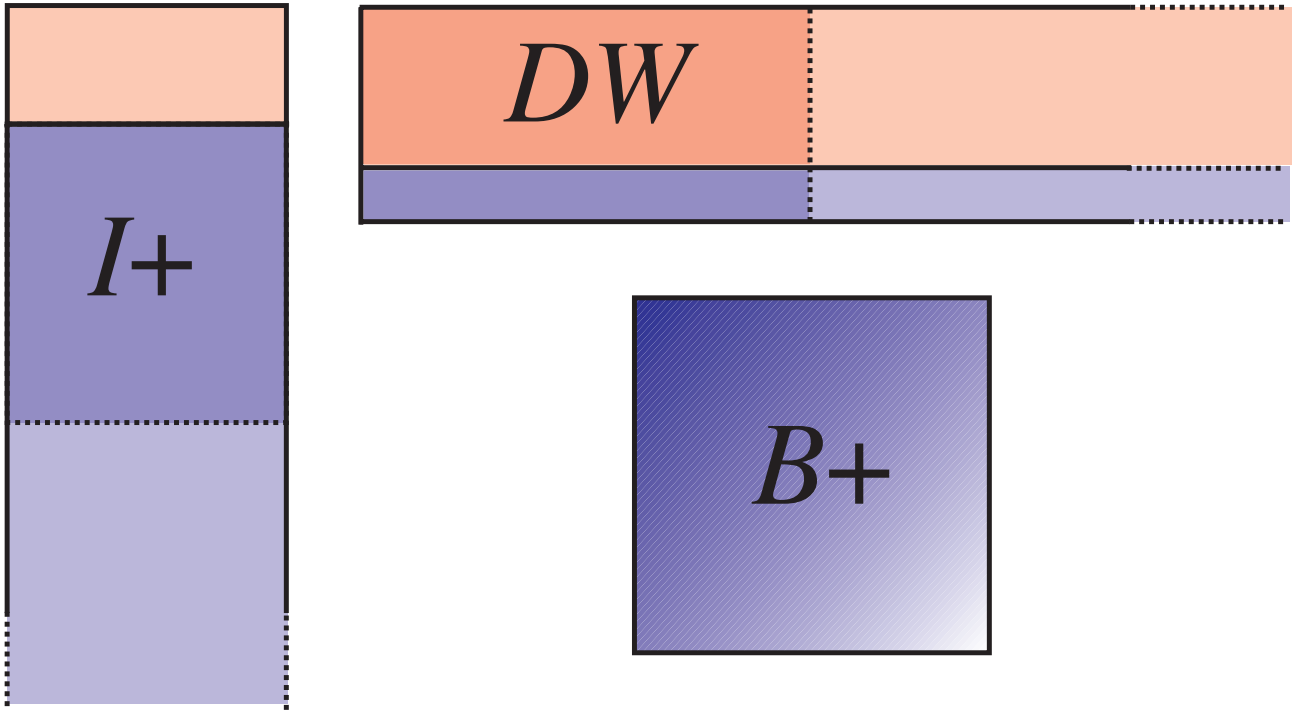
$$x_{ij}^{ks} \leq z_{ij}^{s} \qquad (i,j) \in A \ , \ k \in K \ , \ s \in S_{ij}$$

- Improved relaxation bound: $v(\bar{B}+) = v(\bar{I}+) = v(DW)$[34]
- 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?

[34] F., Gendron "0-1 reformulations of the multicommodity capacitated network design problem" *Disc. App. 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

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

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

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

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

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

- 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

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

- Assumption 2: padding with zeroes

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

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

- 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

$$(\Pi_{\mathcal{B}}) \qquad \max \{ \, cx \; : \; Ax = b \, , \; x = C_{\mathcal{B}}\theta_{\mathcal{B}} \, , \; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \, \} \tag{4}$$

$\equiv$ structured model

$$f_{\mathcal{B}}(y) \; = \; \max\{ \, (c - yA)x + xb \; : \; x = C_{\mathcal{B}}\theta_{\mathcal{B}} \, , \; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \, \} \tag{5}$$

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

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

- Easy[34] 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` $v^* < c\bar{x} + y^*(b - A\bar{x})$

- Easy[34] 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

Structured DW in the literature

- Closely related to the variable redefnition approach[35], except that it offers a dynamic way for solving the LPs

Structured DW in the literature

- Closely related to the variable redefnition approach[35], except that it offers a dynamic way for solving the LPs
- Substantially different from Extended DW[36]

$$U = \left\{ \; u \in \mathbb{N}^n \; : \; Bu \leq d \; \right\} =$$

$$= \left\{ \; w \in \mathbb{R}^n \; : \; w = \sum_{\bar{u} \in U} \bar{u} \theta_{\bar{u}} \; , \; \sum_{\bar{u} \in U} \theta_{\bar{u}} = 1 \; , \; \theta_{\bar{u}} \in \{0, 1\} \quad \bar{u} \in U \; \right\}$$

Structured DW in the literature

- Closely related to the variable redefnition approach[35], except that it offers a dynamic way for solving the LPs
- Substantially different from Extended DW[36]

$$U = \left\{ \; u \in \mathbb{N}^n \; : \; Bu \leq d \; \right\} =$$

$$= \left\{ \; w \in \mathbb{R}^n \; : \; w = \sum_{\bar{u} \in U} \bar{u} \theta_{\bar{u}} \; , \; \sum_{\bar{u} \in U} \theta_{\bar{u}} = 1 \; , \; \theta_{\bar{u}} \in \{0, 1\} \quad \bar{u} \in U \; \right\}$$

- A simple column-and-row generation approach[37]
- Known possible applications (besides FC-MMCF[34]): cutting stock[38,39], time-constrained routing[40], possibly many more

[35] Kipp Martin "Generating alternative mixed-integer programming models using variable redefinition" *OR*, 1987

[36] Vanderbeck "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching [...]" *OR*, 2000

[37] Muter, Birbil, Bülbül "Simultaneous column-and-row generation for [...] column-dependent-rows" *Math. Prog.*, 2012

[38] J.M. Valério de Carvalho "LP models for bin packing and cutting stock problems" *DAM*, 2002

[39] Zak "Row and column generation technique for a multistage cutting stock problem" *C&OR*, 2002

[40] Avella, D'Auria, Salerno "A LP-based heuristic for a time-constrained routing problem" *EJOR*, 2006

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, but stabilizing SDW seems necessary

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:

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

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

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

  dual optimal variables of "$s = b - Ax$" still give d^*, ...

- Move $\bar{y}$, handle t, handle $\mathcal{B}$: as in [4] (or simpler, $\mathcal{B}$ is finite)

- Even better computational results in the right application[41]

[41] F., Gendron "A Stabilized Structured Dantzig-Wolfe Decomposition Method" *Math. Prog.*, 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^2DW, 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^2DW, 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^2DW, 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 $\bar{f}_{\mathcal{B}}$

$$\max \begin{cases} cx + (1-\rho)cx^* + \bar{y}s - \mathcal{D}^*(-s) \\ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ s = Ax + (1-\rho)Ax^* - b \ , \ \ \rho \in [0,1] \end{cases} \tag{6}$$

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^2DW, 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 $\bar{f}_{\mathcal{B}}$

$$\max \begin{cases} cx + (1-\rho)cx^* + \bar{y}s - \mathcal{D}^*(-s) \\ x = C_{\mathcal{B}}\theta_{\mathcal{B}} , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ s = Ax + (1-\rho)Ax^* - b \ , \ \ \rho \in [0,1] \end{cases} \tag{6}$$

  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^2DW
- 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

Structured Decomposition for Multicommodity Flows

- Another telling example: flow relaxation of Multicommodity Flows
- Surprisingly simple $x = C\theta$: use the node-arc formulation
- Start with "empty graph", find paths: if a node/arc is missing, add it
  $\equiv$ "hybrid" 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 (will be included in `BundleSolver 2.0`, one day)
- 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
- 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 ($\eta = *$)

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

Incremental, Inexact, Asynchronous

Incremental Computation of Subproblems

- (Partial) aggregation can contribute to reducing master problem cost but subproblem cost remains the same
- Subproblem cost high if $|K|$ large and/or subproblems hard, trade-off very application-dependent (you get to meet all sorts)
- Clearly interesting to avoid "unnecessary" subproblems computations
- In fact quite easy to understand early on if $f(\bar{y} + d^*) \not\ll f(\bar{y})$ "null steps" can be declared without computing all subproblems[42]
- Early declaring "serious steps" harder, but possible[23] provided you can estimate the Lipchitz constant (nontrivial but doable)
- Fully supported by `BundleSolver`
- Trade-off still all to explore.

[62] Gaudioso, Giallombardo, Miglionico "On Solving the Lagrangian Dual [...] via an Incremental Approach" *COAP*, 2007

Inexact Computation of Subproblems

- Turns out incremental special case of inexact: $f(\bar{y} + d^*)$ only approximately computed
- Powerful general theory well-understood for proximal[23,43] and level[44]
- May require "noise reduction steps": t/l changed without oracle calls (exploit stabilization to sample the space, like "curved search"[45])
- Different noise reductions depending on oracle "unfaithfulness"[23]
- Explicitly provide upper/lower bounds and accuracy to oracle[23]
- Fully supported by `BundleSolver` and `C05Function`
- Can significantly improve total running time, but:
  - details depend on stabilization employed
  - trade-off with number of iterations nontrivial

[43] de Oliveira, Sagastizábal, C. Lemaréchal "Convex Proximal Bundle Methods [...] for Inexact Oracles" *Math. Prog.*, 2014
[44] de Oliveira, Sagastizábal "Level Bundle Methods for Oracles With On-Demand Accuracy" *OM&S*, 2014

Asynchronous Computation of Subproblems

- Clear avenue to reduce wall-clock time: parallelize subproblems
- Master-slave version "obvious" [30], popular for stochastic programs[46]
- Fully supported by `ParallelBundleSolver` via `compute_async()`
- Runs afoul of Amdahl's Law: speedup limited by master problem cost and large master problems is what works best (most often)
- May use specialised algorithms[27] and hardware[30], but issue remains
- Completely asynchronous versions possible[47,48]
- Interesting variants for "loosely coupled subproblems" [49]
- Still work to do, general efficient implementations highly nontrivial

[46] Lubin, Martin, Petra, Sandıkçı "On Parallelizing Dual Decomposition in Stochastic Integer Programming" *O.R. Lett.*, 2013

[47] Iutzeler, Malick, de Oliveira "Asynchronous Level Bundle Methods" *Math. Prog.*, 2020

[48] Fischer "An Asynchronous Proximal Bundle Method" *OO*, 2022

[49] Fischer, Helmberg "A Parallel Bundle Framework for Asynchronous Subspace Optimisation [...]" *SIOPT*, 2014

Conclusions

(Part III)

Conclusions (Part III)

- Decomposition methods (D-W, Benders') old ideas, well-understood, but by-the-book decomposition often not effective enough
- Many nontrivial ideas to improve on the standard approaches

Conclusions (Part III)

- Decomposition methods (D-W, Benders') old ideas, well-understood, but by-the-book decomposition often not effective enough
- Many nontrivial ideas to improve on the standard approaches
- Reduce iterations count: "large" master problems to quickly get the "combinatorial tail" $\Longrightarrow$
  - large master problem time go against Amdhal's Law
  - "unstructured" master problems $\Longrightarrow$ can't use "easy" specialised methods[27] (but there may be ways[50], some structure is there)
- Reduce subproblems cost: incremental/inexact/asynchronous solution + all possible reoptimization
- In all cases, complex multilevel parallel implementation

Conclusions (Part III)

- Decomposition methods (D-W, Benders') old ideas, well-understood, but by-the-book decomposition often not effective enough
- Many nontrivial ideas to improve on the standard approaches
- Reduce iterations count: "large" master problems to quickly get the "combinatorial tail" $\Longrightarrow$
  - large master problem time go against Amdhal's Law
  - "unstructured" master problems $\Longrightarrow$ can't use "easy" specialised methods[27] (but there may be ways[50], some structure is there)
- Reduce subproblems cost: incremental/inexact/asynchronous solution + all possible reoptimization
- In all cases, complex multilevel parallel implementation
- SMS++ aiming for a big slice of this cake (few others seem to want a bite)

Conclusions (Part III)

- Decomposition methods (D-W, Benders') old ideas, well-understood, but by-the-book decomposition often not effective enough
- Many nontrivial ideas to improve on the standard approaches
- Reduce iterations count: "large" master problems to quickly get the "combinatorial tail" $\Longrightarrow$
  - large master problem time go against Amdhal's Law
  - "unstructured" master problems $\Longrightarrow$ can't use "easy" specialised methods[27] (but there may be ways[50], some structure is there)
- Reduce subproblems cost: incremental/inexact/asynchronous solution + all possible reoptimization
- In all cases, complex multilevel parallel implementation
- SMS++ aiming for a big slice of this cake (few others seem to want a bite)
- Lots of fun to be had, all contributions welcome

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