One of Bernard's life-long (scientific) love stories: playing ping-pong between (multicommodity flow) models and (decomposition) algorithms

Antonio Frangioni Bernard Gendron Enrico Gorgone

Dipartimento di Informatica, Università di Pisa

Hommage au Professeur Bernard Gendron

Montreal, February the 23rd, 2023

It all started with the classical Multicommodity flow model

- Graph $G = (N, A)$, classical Multicommodity flow model

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

$$\sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k \qquad i \in N \,,\; k \in K \tag{2}$$

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

$$0 \leq x_{ij}^k \leq u_{ij}^k \qquad (i,j) \in A \,,\; k \in K \tag{4}$$

- Often $b_i^k \equiv (s^k \,,\, t^k \,,\, d^k)$, i.e., commodities $K \equiv$ O-D pairs, possibly with $x_{ij} \to d^k x_{ij}$, $x_{ij} \in \{\, 0 \,,\, 1 \,\}$ (unsplittable routing)
- Pervasive structure in logistic and transportation, often very large (time-space $\Longrightarrow$ acyclic) G, "few" commodities
- Common in many other areas (telecommunications, energy, ...), possibly "small" (undirected) G, "many" commodities
- Interesting links with many hard problems (e.g. Max-Cut)
- "Hard" even if continuous: very-large-scale LPs

The paradise of decomposition

- Many sources of structure $\Longrightarrow$ the paradise of decomposition[1,2]
- Lagrangian relaxation[3] of linking constraints:
  - (3) $\Longrightarrow$ flow (shortest path) relaxation
  - (2) $\Longrightarrow$ knapsack relaxation
  - others possible (will see)
- Benders' decomposition[4] of linking variables:
  - Linking variables can be artificially added (resource decomposition)[5]

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

- I did mostly Lagrange, but many ideas can be applied to Benders[6] and Bernard did work on Benders (for network design, will see)[7]

[1] Ford, Fulkerson "A Suggested Computation for Maximal Multicommodity Network Flows" *Man. Sci.*, 1958
[2] Dantzig, Wolfe "The Decomposition Principle for Linear Programs" *Op. Res.*, 1960
[3] Geoffrion "Lagrangean relaxation for integer programming" *Math. Prog. Study*, 1974
[4] Benders "Partitioning procedures for solving mixed-variables programming problems" *Num. Math.*, 1962
[5] Kennington, Shalaby "An Effective Subgradient Procedure for Minimal Cost Multicomm. Flow Problems" *Man. Sci.* 1977
[6] van Ackooij, F., de Oliveira "Inexact Stabilized Benders' Decomposition Approaches, with Application [...]" *CO&A*, 2016
[7] Costa, Cordeau, Gendron "Benders, metric and cutset inequalities for multicommodity [...] network design" *CO&A*, 2009

(Dantzig-Wolfe) Decomposition 101

- The general form of structure we consider:

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

$Ax = b$ "complicating" $\equiv$ optimizing upon X "easy" $\equiv$ convex

- Almost always $X = \bigotimes_{h \in \mathcal{K}} X^h$ ($\mathcal{K} \neq K$) $\equiv$ $Ax = b$ linking constraints

- Our X compact, represent it by vertices (otherwise just add extreme rays)

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

$\Longrightarrow$ Dantzig-Wolfe reformulation[2] of (Π):

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

- X nonconvex $\Longrightarrow$ solving "best" convex relaxation

$$(\bar{\Pi}) \qquad \max \{ \, cx \; : \; Ax = b \, , \; x \in conv(\, X \,) \, \} \tag{5}$$

D-W decomposition $\equiv$ Lagrangian relaxation

- $\mathcal{B} \subset X$ (small), solve master problem restricted to $\mathcal{B}$

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

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

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

  (Lagrangian relaxation), optimal solution $\bar{x}$ of $(\Pi_{\lambda^*}) \rightarrow \mathcal{B}$

- Dual: $(\Delta_{\mathcal{B}}) \;\; \min \{ \, f_{\mathcal{B}}(\, \lambda \,) = \max \{ \, cx + \lambda(b - Ax) \; : \; x \in \mathcal{B} \, \} \, \}$

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

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

  $\Longrightarrow (\Delta_{\mathcal{B}})$ outer approximation of Lagrangian dual $\equiv (\Pi)$

  $$(\Delta) \qquad \min \{ \, f(\, \lambda \,) = \max \{ \, cx + \lambda(b - Ax) \; : \; x \in X \, \} \, \} \qquad (6)$$

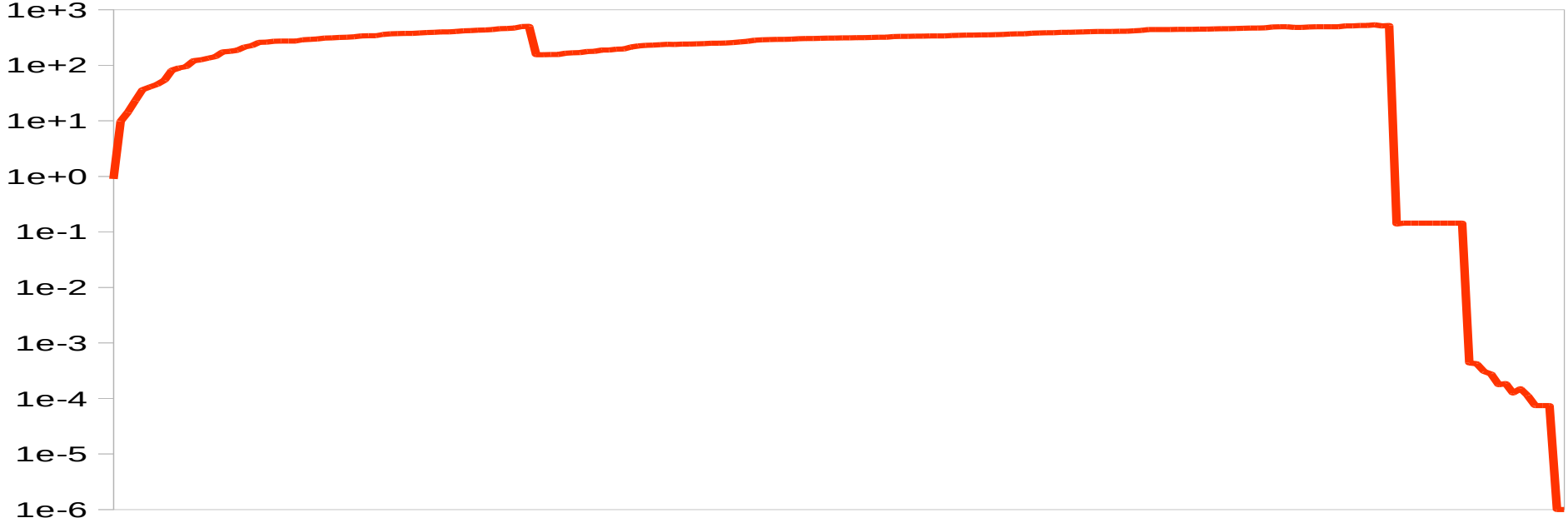
- Dantzig-Wolfe decomposition $\equiv$ Cutting Plane approach to (Δ)[8]

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

All well and nice, but does it work well?

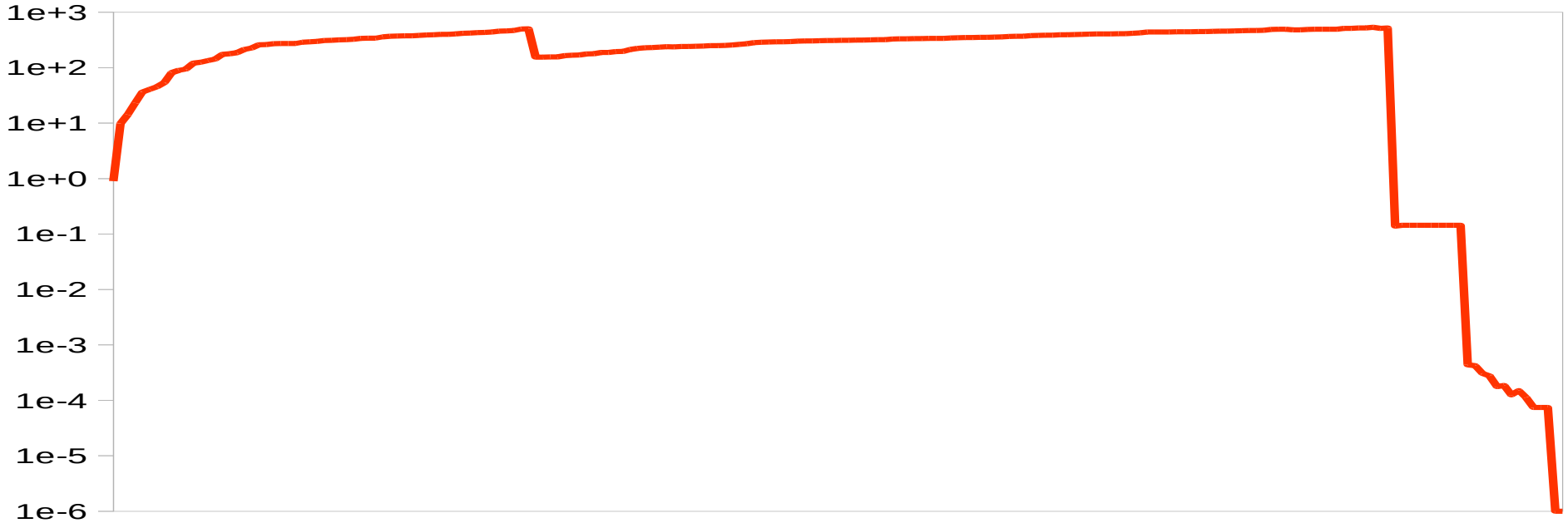
All well and nice, but does it work well?

- By-the-book? Not really



All well and nice, but does it work well?

- By-the-book? Not really



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

How to deal with instability

- λ^*_{k+1} can be very far from λ^*_k, where $f_{\mathcal{B}}$ is a "bad model" of f
- If $\{ \lambda^*_k \}$ is unstable, then stabilize it around stability centre $\bar{\lambda}$
- Stabilizing term $\mathcal{D}_t$ with parameter t, stabilized master problems

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

$$(\Pi_{\mathcal{B},\bar{\lambda},\mathcal{D}_t}) \ \max \left\{ cx + \bar{\lambda}(b - Ax) - \mathcal{D}^*_t(Ax - b) \ : \ x \in conv(\mathcal{B}) \right\}$$

("*" = Fenchel's conjugate): a generalized augmented Lagrangian

- Change $\bar{\lambda}$ when $f(\bar{\lambda} + d^*) \ll f(\bar{\lambda})$, appropriate $\mathcal{D} \Longrightarrow$ converges[9]
- Choosing t nontrivial
- Aggregation trick: right $\mathcal{D} \Longrightarrow$ still converges with "poorman bundle" $\mathcal{B} = \{ x^* \}$ (although rather slowly[10] $\approx$ volume[11] $\equiv$ subgradient)

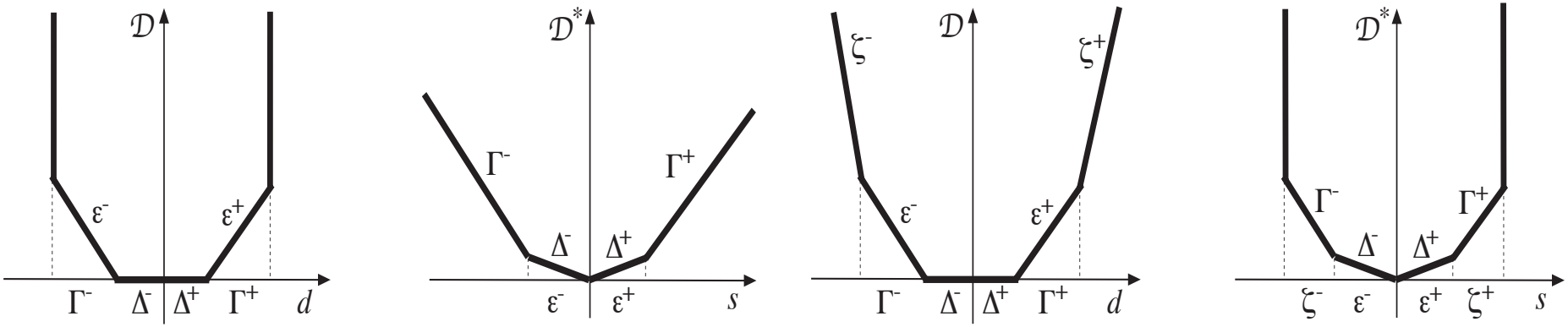
[9] F. "Generalized Bundle Methods" *SIOPT*, 2002

[10] Briant, Lemaréchal, et. al. "Comparison of bundle and classical column generation" *Math. Prog.*, 2006

[11] Bahiense, Maculan, Sagastizábal "The volume algorithm revisited: relation with bundle methods" *Math. Prog.*, 2002

What is an appropriate stabilization?

- Simplest: $\mathcal{D}_t \equiv \| d \|_\infty \leq t$, $\mathcal{D}_t^* = t\| \cdot \|_2^2$ ("boxstep")[12]
- Better[13]: $\mathcal{D}_t = \frac{1}{2t}\| \cdot \|_2^2$, $\mathcal{D}_t^* = \frac{1}{2}t\| \cdot \|_2^2$ (may use specialized QP solvers[14])
- Keep LP master: piecewise-linear approximations[15]



- Several other ideas[16] (level stabilization, centres, better "Hessian", ...)

---

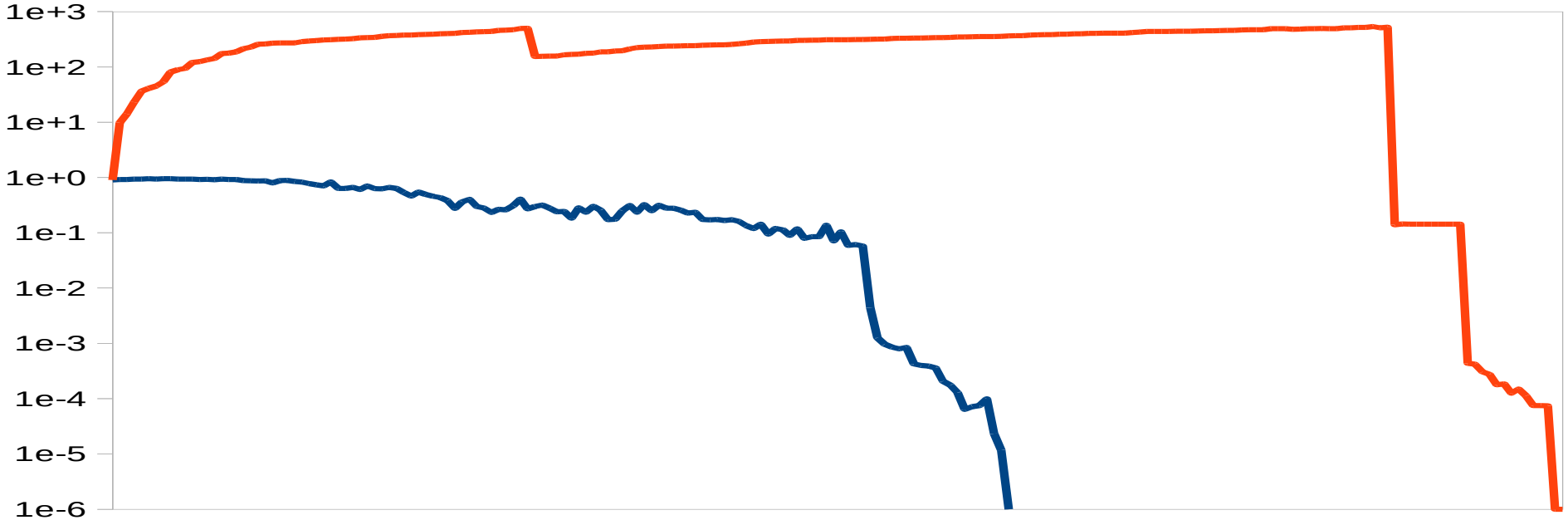
[12] Marsten, Hogan, Blankenship "The Boxstep Method for Large-scale Optimization" *OR*, 1975
[13] Lemaréchal "Bundle Methods in Nonsmooth Optimization" in *Nonsmooth Optimization* vol. 3, 1978
[14] F. "Solving semidefinite quadratic problems within nonsmooth optimization algorithms" *Computers & O.R.*, 1996
[15] Ben Amor, Desrosiers, F. "On the choice of explicit stabilizing terms in column generation" *Disc. Appl. Math.*, 2009
[16] F., "Standard Bundle Methods: Untrusted Models and Duality" in *Numerical Nonsmooth Optimization: ...*, 2020

All well and nice, but does it work well?

[17] Nemirovsky, Yudin "Problem Complexity and Method Efficiency in Optimization" Wiley, 1983

All well and nice, but does it work well?

- It depends on what "well" means, but surely better



- Black-box nonsmooth optimization is $\Omega(1/\varepsilon^2)$ in general[17]
- Convergence slow-ish (but at lest some) until mysterious threshold hit
- At least, better information accrued sooner $\Longrightarrow$ "quick tail" starts sooner
- Can make a huge difference in applications

[17] Nemirovsky, Yudin "Problem Complexity and Method Efficiency in Optimization" Wiley, 1983

Indeed, it worked well enough for Multicommodity flows

k	n	m	b	*Size*	MMCFB	Cplex	PPRN	IPM
4	64	362	148	1.4e+3	0.07	0.22	0.13	1.44
8	64	371	183	3.0e+3	0.26	0.50	0.52	4.26
16	64	356	191	5.7e+3	1.08	2.01	3.41	16.03
32	64	362	208	1.2e+4	3.42	12.99	22.04	43.27
64	64	361	213	2.3e+4	8.53	115.99	147.10	114.19
4	128	694	293	2.8e+3	0.58	0.54	0.85	6.45
8	128	735	363	5.9e+3	2.57	1.81	4.79	26.32
16	128	766	424	1.2e+4	11.30	17.31	40.57	116.26
32	128	779	445	2.5e+4	27.72	212.09	503.48	346.91
64	128	784	469	5.0e+4	44.04	1137.05	2215.48	719.69
128	128	808	485	1.0e+5	52.15	5816.54	6521.94	1546.91
4	256	1401	570	5.6e+3	7.54	2.38	9.88	51.00
8	256	1486	743	1.2e+4	25.09	15.48	105.89	208.10
16	256	1553	854	2.5e+4	60.85	180.06	955.20	844.09
32	256	1572	907	5.0e+4	107.54	1339.46	6605.45	1782.47
64	256	1573	931	1.0e+5	144.75	7463.14	18467.73	3441.62
128	256	1581	932	2.0e+5	223.13	35891.37	61522.94	9074.31
256	256	1503	902	3.8e+5	445.81	110897+	187156+	17279.00

- We could handily beat the state-of-the-art Cplex 3.0 and others[18]

[18] F., Gallo "A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems" *IJoC*, 1999

[19] Cappanera, F. "[...] Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems" *IJoC*, 2003

Indeed, it worked well enough for Multicommodity flows

Group	T_1	$s\%$	T_4	T_{16}	T_{64}
64-64	21.31	1.10	5.98	2.08	1.00
128-64	123.66	1.25	35.70	13.16	7.01
128-128	159.78	0.66	42.04	12.65	4.95
256-64	466.35	1.51	129.75	44.69	21.89
256-128	718.35	0.62	188.96	57.23	22.99
256-256	1404.48	0.30	348.46	98.30	33.85
512-512	15898.89	0.22	$*$	1025.26	291.40

- We could handily beat the state-of-the-art Cplex 3.0 and others[18]
- We could even parallelise on a supercomputer with a whopping 64 CPU[19]

[18] F., Gallo "A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems" *IJoC*, 1999

[19] Cappanera, F. "[...] Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems" *IJoC*, 2003

Indeed, it worked well enough for Multicommodity flows

Group	T_1	$s\%$	T_4	T_{16}	T_{64}
64-64	21.31	1.10	5.98	2.08	1.00
128-64	123.66	1.25	35.70	13.16	7.01
128-128	159.78	0.66	42.04	12.65	4.95
256-64	466.35	1.51	129.75	44.69	21.89
256-128	718.35	0.62	188.96	57.23	22.99
256-256	1404.48	0.30	348.46	98.30	33.85
512-512	15898.89	0.22	$*$	1025.26	291.40

- We could handily beat the state-of-the-art Cplex 3.0 and others[18]
- We could even parallelise on a supercomputer with a whopping 64 CPU[19]
- But this was not enough for Bernard . . .

[18] F., Gallo "A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems" *IJoC*, 1999

[19] Cappanera, F. "[. . .] Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems" *IJoC*, 2003

...for he wanted to solve Multicommodity Network Design

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{7}$$

$$\sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k \qquad i \in N \,,\; k \in K \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} \qquad (i,j) \in A \tag{8}$$

$$0 \leq x_{ij}^k \leq u_{ij}^k y_{ij} \qquad (i,j) \in A \,,\; k \in K \tag{9}$$

$$y \in Y \subseteq \{ 0 \,,\, 1 \}^m \tag{10}$$

- Reasonably good bounds but only with strong forcing constraints (9)
- Just one more subproblem, but a lot more constraints (9) to relax $\equiv$ much larger dual space (harder) and much more costly master problem
- In fact, relaxing (2) (knapsack relaxation) competitive: less multipliers (but unconstrained), still (arc) decomposable if $Y = \{ 0 \,,\, 1 \}^m$
- Flow relaxation requires dynamic bundle methods[20], many other uses[21]

[20] Belloni, Sagastizábal "Dynamic bundle methods" *Math. Prog.*, 2009

[21] F., Lodi, Rinaldi "New approaches for optimizing over the semimetric polytope" *Math. Prog.*, 2005

Which worked well, sort of

Problems	CPXW	WB	CPXS	SS	SB	KS	KB
25,100,10	2.3e-1	2.3e-1	0.0	5.3e-4	1.8e-4	7.6e-4	2.7e-4
(3)	0.1	0.0	1.3	1.1	1.0	0.6	0.8
25,100,30	2.2e-1	2.2e-1	0.0	4.0e-4	1.4e-4	9.8e-4	5.5e-4
(3)	0.6	0.2	11.3	3.0	3.5	1.2	2.3
100,400,10	2.8e-1	2.8e-1	0.0	1.1e-3	6.7e-4	1.6e-3	1.3e-3
(3)	0.3	0.1	35.9	4.2	4.8	1.7	3.0
100,400,30	2.9e-1	2.9e-1	0.0	1.0e-3	1.1e-3	1.9e-3	2.5e-3
(3)	5.9	2.3	351.9	14.3	16.7	4.5	9.1

- Issue: $>$ 10-100 subgradients filled our mighty 64Mb (not a typo) of RAM $\Longrightarrow$ never really got to the "fast tail" convergence
- Yet bundle competitive with subgradient, flow and knapsack traded blows, 1e−5 to 1e−3 accuracy good enough for a B&B[22]
- Could have been better, still my most cited article ever[23]

[22] Holmberg, Yuan "A Lagrangian [...] B&B Approach for the Capacitated Network Design Problem" *Op. Res.*, 2000
[23] Crainic, F., Gendron "Bundle-based Relaxation Methods for Multicommodity [...] Network Design Problems" *DAM*, 2001

But Bernard was not happy, so we kept pushing

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

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

- Exploit separability: $X = X^1 \times X^2 \times \ldots \times X^{|K|} \Longrightarrow$
  $conv(X) = conv(X^1) \times conv(X^2) \times \ldots \times conv(X^{|K|}) \Longrightarrow$
  a different θ_s^k for each $\bar{x}^{k,s}$ (aggregated $\equiv \theta_s^k = \theta_s^h, h \neq k$, innatural)
- Simple scaling leads to arc-path formulation (in O-D case):
  $p \in \mathcal{P}^k = \{ s^k\text{–}t^k$ paths $\}$, c_p cost, $f_p (= d^k \theta_s^k)$ flow, $\mathcal{P} = \cup_{k\in K} \mathcal{P}^k$

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

Disaggregated decomposition

- Disaggregated formulation: more columns but sparser, more rows
- Master problem size $\approx$ time increases, but convergence speed increases $\equiv$ consistent improvement if you have enough RAM
- Much more efficient for Multicommodity Flows[24] and others[25]
- But not for Network Design! So we had to understand why

[24] Jones, Lustig, et. al. "Multicommodity Network Flows: The Impact of Formulation on Decomposition" *Math. Prog.*, 1993
[25] Borghetti, F., Lacalandra, Nucci "Lagrangian Heuristics Based on Disaggregated Bundle [...]" *IEEE TPWRS*, 2003

How not to do disaggregated decomposition

- $\mathcal{S}$ = extreme points of y ($2^{|A|}$ vertices of the unitary hypercube):

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

How not to do disaggregated decomposition

- $\mathcal{S}$ = extreme points of y ($2^{|A|}$ vertices of the unitary hypercube):

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

- Is this sane? Arguably not: replacing a $2n$ formulation with a 2^n one!
- The problem on y variables is too easy, do not D-W it

How not to do disaggregated decomposition

- $\mathcal{S}$ = extreme points of y ($2^{|A|}$ vertices of the unitary hypercube):

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

- Is this sane? Arguably not: replacing a $2n$ formulation with a 2^n one!
- The problem on y variables is too easy, do not D-W it
- Or D-W it more: $\{\,0\,,\,1\,\}^m$ is a Cartesian product: why not $\mathcal{S}^{ij} = \{0,1\}$?
- $y_{ij} \quad \longrightarrow \quad 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1} \;\;,\;\; \theta^{ij,0} + \theta^{ij,1} = 1 \;\;,\;\; \theta^{ij,0} \geq 0 \;\;,\;\; \theta^{ij,1} \geq 0$

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

How not to do disaggregated decomposition

- $\mathcal{S}$ = extreme points of y ($2^{|A|}$ vertices of the unitary hypercube):

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

- Is this sane? Arguably not: replacing a $2n$ formulation with a 2^n one!
- The problem on y variables is too easy, do not D-W it
- Or D-W it more: $\{\,0\,,\,1\,\}^m$ is a Cartesian product: why not $\mathcal{S}^{ij} = \{0,1\}$?
- $y_{ij} \quad \longrightarrow \quad 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1} \;\;,\;\; \theta^{ij,0} + \theta^{ij,1} = 1 \;\;,\;\; \theta^{ij,0} \geq 0 \;\;,\;\; \theta^{ij,1} \geq 0$

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

(no, ... really?!)

How to do a disaggregated decomposition

- Arc-path formulation with original arc design variables

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

  only generate the right variables, those that are too many

- But if one had (say) $\sum_{(i,j) \in A} y_{ij} \leq r$: a linking constraint in Y
  $\Longrightarrow$ the design subproblem can no longer be disaggregated
- Yet, one could just add that constraint to the master problem
- Can this be stabilized? Of course it can[26]

[26] F., Gorgone "Bundle methods for sum-functions with "easy" components: [...] network design" *Math. Prog.*, 2013

Stabilization with easy components

- Required structure: X^1 arbitrary, X^2 has compact convex formulation
  $(\Pi)\quad \max\left\{\ c_1x_1 + c_2(x_2)\ :\ x_1 \in X^1\ ,\ G(x_2) \le g\ ,\ A_1x_1 + A_2x_2 = b\ \right\}$
- Lagrangian function $f(\,\lambda\,) = f^1(\,\lambda\,) + f^2(\,\lambda\,)\,(-\lambda b)$, two components
- Primal master problem: "just plug in the easy set"

$$(\Pi_{\mathcal{B}})\ \max\begin{cases} c_1x_1 + c_2(x_2) \\ A_1x_1 - A_2x_2 = b \\ x_1 \in conv(\mathcal{B})\ ,\ x_2 \in X^2 \end{cases} \equiv \max\begin{cases} c_1\left(\sum_{\bar{x}_1\in\mathcal{B}} \bar{x}_1\theta_{\bar{x}_1}\right) + c_2(x_2) \\ A_1\left(\sum_{\bar{x}_1\in\mathcal{B}} \bar{x}_1\theta_{\bar{x}_1}\right) + A_2x_2 = b \\ \sum_{\bar{x}_1\in\mathcal{B}} \theta_{\bar{x}_1} = 1\quad ,\quad G(x_2) \le g \end{cases}$$

- Dual master problem: $(\Delta_{\mathcal{B}})\quad \min\left\{\ \lambda b + f^1_{\mathcal{B}}(\,\lambda\,) + f^2(\,\lambda\,)\ \right\}$
  i.e., insert "full" description of f^2 in the master problem
- Larger master problem at the beginning, but "perfect" information known
- Of course, stabilization + multiple easy/hard components ...

All well and nice, but does it work well?

All well and nice, but does it work well?

- You bet, but you have to do it right: let information accumulate
- Fast tail starts immediately if $\geq$ 50000 subgradients + no harsh removals

All well and nice, but does it work well?

- You bet, but you have to do it right: let information accumulate
- Fast tail starts immediately if $\geq$ 50000 subgradients + no harsh removals

Cplex	easy		aggregate			volume		
dual	1e-6	1e-12	time	it	gap	time	it	gap
39	26	32	322	10320	1e-6	6	871	8e-3
132	28	56	294	5300	1e-6	12	831	9e-3
301	21	26	5033	27231	1e-6	26	794	3e-3
1930	133	133	3122	14547	1e-6	51	760	4e-2
131	2	3	344	7169	1e-6	12	827	3e-3
708	246	337	2256	17034	2e-5	29	869	1e-2
2167	284	508	5475	15061	3e-6	58	817	2e-2
8908	242	253	11863	13953	1e-6	109	765	2e-2

- Much better accuracy/time than `Cplex` and competing decompositions
- Finally competitive even for Network Design, very happy
- Of course, meanwhile Barnard had already moved on

Knapsack decomposition for Network Loading

- y general integers, relax flow conservation constraints (2)

$$
\begin{aligned}
\min \textstyle\sum_{(i,j)\in A} \big(\sum_{k\in K} (d^k c_{ij}^k - \pi_i^k + \pi_j^k) x_{ij}^k + f_{ij} y_{ij} \big) & \\
\textstyle\sum_{k\in K} d^k x_{ij}^k \leq u_{ij} y_{ij} & \qquad (i,j) \in A \\
x_{ij}^k \in [0,1] & \qquad (i,j) \in A,\ k \in K \\
y_{ij} \in \mathbb{N} & \qquad (i,j) \in A
\end{aligned}
$$

- 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[27]

$$
\begin{gathered}
a_k = d^k / u_{ij} \qquad a(S) = \textstyle\sum_{k\in S} a_k \qquad S \subseteq K \\
\textstyle\sum_{k\in S} a_k (1 - x_{ij}^k) \geq (\, a(S) - \lfloor a(S) \rfloor \,)(\, \lceil a(S) \rceil - y \,)
\end{gathered}
\tag{11}
$$

- $\bar{I}+$ = continuous relaxation of (1)–(10) + (11) $\equiv$ DW[28]

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

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

RG vs. StabDW, strange game: the only winning move . . .

- Large difficult instances, lightly ($C = 1$) to tightly ($C = 16$) capacitated
- Aggregated and/or non-stabilised DW too slow, only Stabilized DW "works" (but $\|\cdot\|_\infty$ stabilization, $\|\cdot\|_2^2$ too costly, see below)

Problem			$I+$		StabDW	
$\|A\|$	C	imp	cpu	it	cpu	it
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

- Trade blows depending on C, but basically both lose

Reformulation III: Binary formulation B

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

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

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

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

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

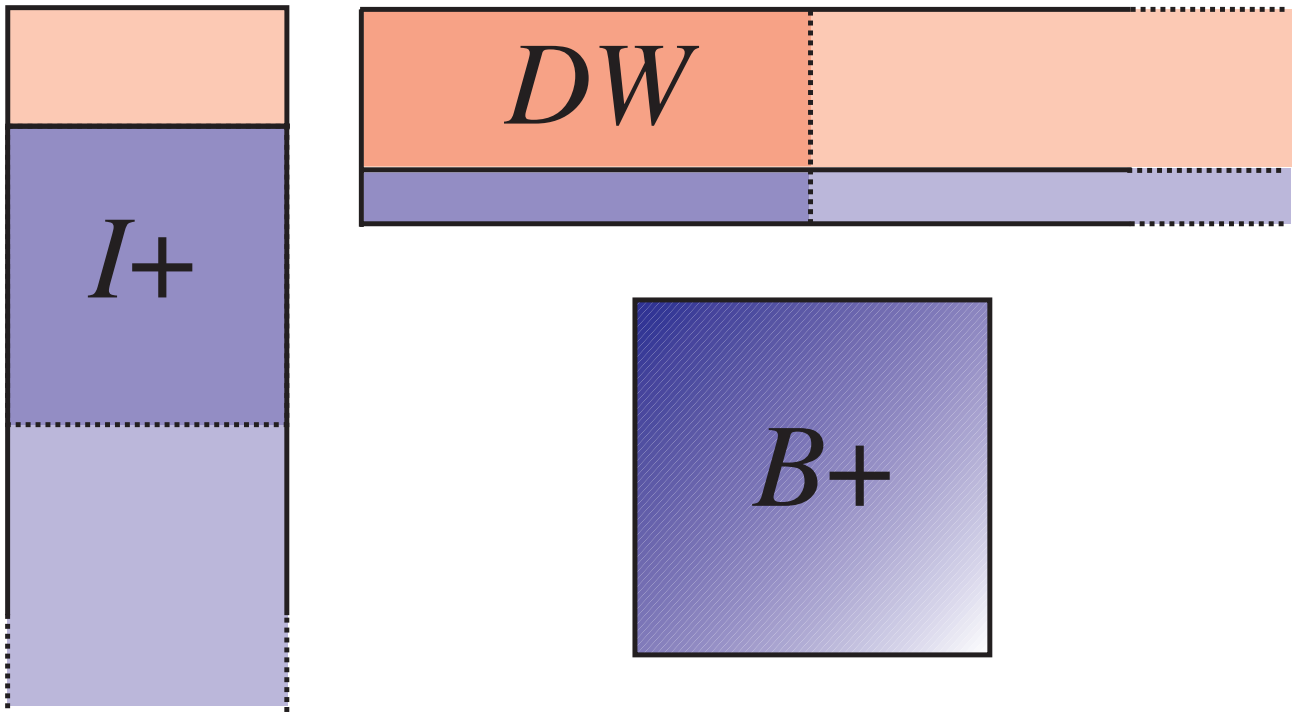
- $+$ extended linking inequalities $x_{ij}^{ks} \leq y_{ij}^s \quad (i,j) \in A \ , \ k \in K \ , \ s \in S_{ij}$

$\Longrightarrow B+$ same bound as $\bar{I}+$ and DW[29]

[29] F., Gendron "0-1 reformulations of the multicommodity capacitated network design problem" *DAM*, 2009

Reformulations, reformulations, reformulations

- In fact, binary formulation describes $conv(\, X^{ij} \,) \equiv$ integrality property $\Longrightarrow$ optimizing over $X \Longrightarrow conv(\, X \,)$ easy
- Pseudo-polynomial number of variables and constraints
- Substantially different from both RG and DW



- Need to generate both rows and columns: how?

The Structured Dantzig-Wolfe Idea

- Assumption 1 (alternative (large) Formulation of "easy" set)
$$conv(\,X\,) = \{\; x = C\theta \;:\; \Gamma\theta \leq \gamma \;\}$$
- Assumption 2 (padding with zeroes): $\Gamma_{\mathcal{B}}\bar{\theta}_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \implies \Gamma\left[\,\bar{\theta}_{\mathcal{B}}\,,\,0\,\right] \leq \gamma$
$$\implies X_{\mathcal{B}} = \left\{\; x = C_{\mathcal{B}}\theta_{\mathcal{B}} \;:\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \;\right\} \subseteq conv(\,X\,)$$
- 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 $\mathcal{B}' \supset \mathcal{B}$ ($\implies \Gamma_{\mathcal{B}'}, \gamma_{\mathcal{B}'}$) such that $\exists\, \mathcal{B}'' \supseteq \mathcal{B}'$ such that $\bar{x} \in X_{\mathcal{B}''}$.
- Structured master problem
$$(\Pi_{\mathcal{B}}) \qquad \max\left\{\; cx \;:\; Ax = b\,,\; x = C_{\mathcal{B}}\theta_{\mathcal{B}}\,,\;\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \;\right\} \tag{12}$$

$\equiv$ structured model

$$f_{\mathcal{B}}(\,\lambda\,) \;=\; \max\{\; (c - \lambda A)x + xb \;:\; x = C_{\mathcal{B}}\theta_{\mathcal{B}}\,,\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \;\} \tag{13}$$

The Structured Dantzig-Wolfe Algorithm

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

- Relatively easy[29] 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

And it does work somewhat better

Problem			$I+$			StabDW		StructDW		
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it
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

- Save sometimes for highly capacitated instances
- Extra advantage: quickly solve reduced binary model to integer optimality ("price and branch") giving better feasible solutions than integer model
- Still likely room for improvement: stabilizing SDW seems promising

Stabilizing the Structured Dantzig-Wolfe Algorithm

- Exactly the same as stabilizing DW: stabilized master problem

$$(\Delta_{\mathcal{B},\bar{y},\mathcal{D}}) \qquad \min \left\{ f_{\mathcal{B}}(\bar{\lambda} + d) + \mathcal{D}(d) \right\} \tag{14}$$

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

- Even simpler from the primal viewpoint[30]:

$$\max \left\{ cx + \bar{\lambda} z - \mathcal{D}^*(-z) \,:\, z = b - Ax \,,\, x = C_{\mathcal{B}}\theta_{\mathcal{B}} \,,\, \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\} \tag{15}$$

- 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[9], actually even somewhat simpler because $\mathcal{B}$ is inherently finite

[30] F., Gendron "A Stabilized Structured Dantzig-Wolfe Decomposition Method" *Math. Prog.*, 2013

And it actually works a lot better

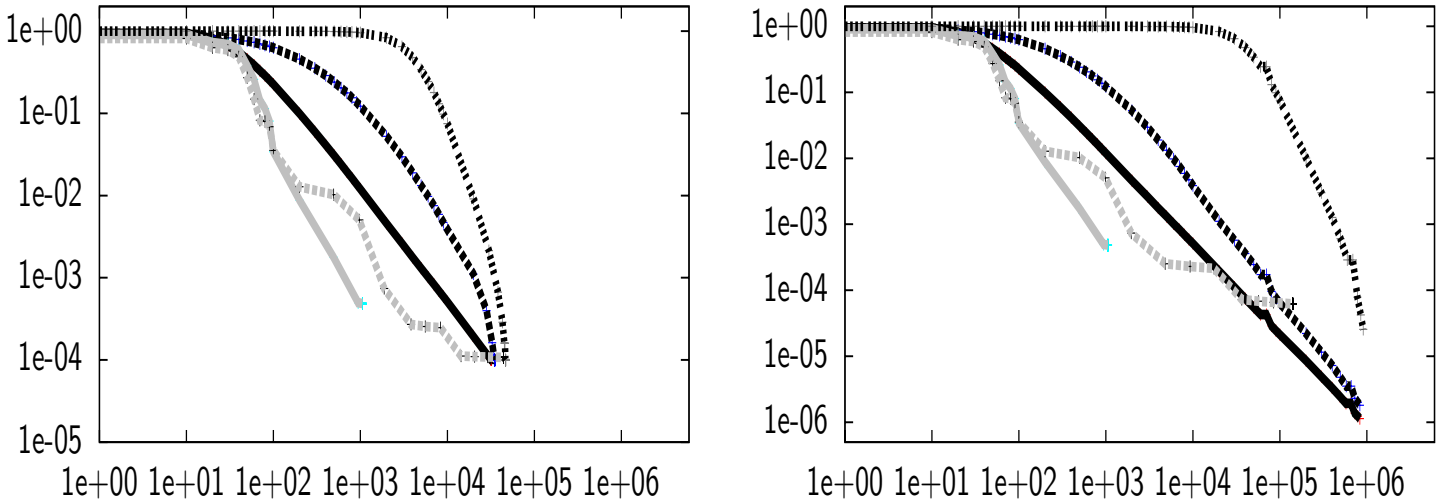
- Can do smart warm-start (MCF + subgradient) to improve performances

	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	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

- Quadratic stabilization converges faster but master problem too costly
- Warm-started stabilised (with $\|\cdot\|_\infty$) structured decomposition gives extremely good upper and lower bounds in (relatively) short time

Not that we entirely gave up on subgradients, either

- In fact we tested them all very thoroughly (for knapsack decomposition)[31]
- We even tested fancy smoothed subgradient ($\equiv$ quadratic knapsack[32]) but results were not good: $\approx$linear in a doubly-logarithmic chart



- Subgradients faster but flatline at $\varepsilon \approx$ 1e−4, smoothed does $\varepsilon =$ 1e−6 but it requires 1e+6 iterations to get there
- Exploiting information about f_* helps (black solid line) but not enough[33]

[31] F., Gendron, Gorgone "On the Computational Efficiency of Subgradient Methods [...]" *Math. Prog. Comp.*, 2017
[32] F., Gorgone "A Library for Continuous Convex Separable Quadratic Knapsack Problems" *EJOR*, 2013
[33] F., Gendron, Gorgone "Dynamic Smoothness Parameter for Fast Gradient Methods" *Opt. Lett.*, 2018

But Bernard loved models more than algorithms

- . . . and was always capable of finding new gems in a highly mined cave

But Bernard loved models more than algorithms

- . . . and was always capable of finding new gems in a highly mined cave
- He took the venerable knapsack relaxation and came up with three new node-based ones by playing nifty reformulation tricks

But Bernard loved models more than algorithms

- ...and was always capable of finding new gems in a highly mined cave
- He took the venerable knapsack relaxation and came up with three new node-based ones by playing nifty reformulation tricks
- $K_i^{O/T/D} = \{\, k \in K \;:\; i \text{ is origin/transhipment/destination for } i \,\}$
- Add redundant $\sum_{j \in N_i^+} x_{ij}^k \leq g_i^k = \min\{\, d^k \,,\, \sum_{j \in N_i^-} u_{ji} \,\} \;\; i \in N \,,\, k \in K_i^T$
- Facility location relaxation, decomposes by $i \in N \equiv$ node:

$$
\begin{array}{lr}
\min \sum_{j \in N_i^+} \sum_{k \in K} c_{ij}^k(\pi) x_{ij}^k + f_{ij} y_{ij} & \\
\sum_{j \in N_i^+} x_{ij}^k = d^k & k \in K_i^O \\
\sum_{j \in N_i^+} x_{ij}^k \leq g_i^k & k \in K_i^T \\
x_{ij}^k = 0 & j \in N_i^+ \,,\, k \in K_i^D \cup K_j^O \\
\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} & j \in N_i^+ \\
0 \leq x_{ij}^k \leq d^k y_{ij} & j \in N_i^+ \,,\, k \in K \\
y_{ij} \in \{0, 1\} & j \in N_i^+
\end{array}
$$

And then another one

- Introduce copies of design (z) and flow (v) variables, then link them with copy constraints (Lagrangian decomposition)

$$z_{ij} - y_{ij} = 0 \qquad (i,j) \in A \tag{16}$$

$$v_{ij}^k - x_{ij}^k = 0 \qquad (i,j) \in A\,,\ k \in K \tag{17}$$

- Add a bunch of redundant constraints

$$\sum_{j \in N_i^-} v_{ji}^k = d^k \qquad i \in N\,,\ k \in K_i^D$$

$$v_{ji}^k = 0 \qquad (j,i) \in A\,,\ k \in K_i^O \cup K_j^D$$

$$\sum_{k \in K} v_{ji}^k \leq u_{ji} z_{ji} \qquad (j,i) \in A$$

$$0 \leq v_{ji}^k \leq d^k z_{ji} \qquad (j,i) \in A\,,\ k \in K$$

$$z_{ji} \in \{0,1\} \qquad (j,i) \in A$$

$$\sum_{j \in N_i^-} v_{ji}^k \leq h_i^k = \min\{\, d^k\,,\ \sum_{j \in N_i^+} u_{ij}\,\} \qquad i \in N\,,\ k \in K_i^T$$

- Now relax (16) and (17) together with (2)

Behold the forward-backward facility location relaxation

- One problem (for each $i \in N$) just like before, except with
  $\min \sum_{j \in N_i^+} \sum_{k \in K} c_{ij}^k(\omega, \pi) x_{ij}^k + f_{ij}(\gamma) y_{ij}$
- The other (for each $i \in N$) analogous on the (v, z)

$$
\begin{aligned}
\min \ & \textstyle\sum_{j \in N_i^-} \sum_{k \in K} c_{ji}^k(\omega) v_{ji}^k + f_{ji}(\gamma) z_{ji} & \\
& \textstyle\sum_{j \in N_i^-} v_{ji}^k = d^k & k \in K_i^D \\
& \textstyle\sum_{j \in N_i^-} v_{ji}^k \leq h_i^k & k \in K_i^T \\
& v_{ji}^k = 0 & j \in N_i^- \,,\ k \in K_i^O \cup K_j^D \\
& \textstyle\sum_{k \in K} v_{ji}^k \leq u_{ji} z_{ji} & j \in N_i^- \\
& z_{ji} \in \{0, 1\} & j \in N_i^-
\end{aligned}
$$

- Still decomposes by $i \in N \equiv$ node, but now two CFL problems
- Correspondingly, better bound than the facility location relaxation

And then yet another one

- Add to the forward-backward facility location relaxation the constraints
$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} v_{ji}^k = 0 \quad i \in N\,,\ k \in K_i^T$$
- Two subproblems $\mapsto$ multicommodity single-node fixed-charge problem more difficult $\Longrightarrow$ better bound than forward-backward relaxation
- A whole new set of bound quality/time trade-offs to explore

	Z^{LP}	Z^{FW}	Z^{KN}	Z^{FL}	Z^{FB}	Z^{SN}
Average gap	—	0.003	0.008	-0.508	-0.919	-1.781
Minimum gap	—	0.000	0.000	-4.767	-7.713	-20.518
Total time (sec.)	170.25	7.40	125.56	699.74	4073.34	4677.71
Number of iterations	—	20	5866	284	373	316
Lagrangian time (%)	—	5	18	28	10	65
Master problem time (%)	—	95	82	72	90	35

- A bunch of new Lagrangian-based math-heuristics, competitive results[34]
- A renewed interest in incremental/inexact Bundle methods[35]
- Lots of fun!

[34] Kazemzadeh, Bektas, Crainic, F., Gendron, Gorgone "Node-Based Lagrangian Relaxations [...]" *DAM*, 2022
[35] van Ackooij, F "Incremental Bundle Methods Using Upper Modelsìì *SIOPT*, 2018

And he was not done with knapsack relaxation either

- Knapsack relaxation decomposes by arc if $Y = \{ 0 , 1 \}^{|A|}$

$$\begin{array}{lll} \min & \sum_{(i,j)\in A} \big(\sum_{k\in K} (c_{ij}^k - \pi_i^k + \pi_j^k) x_{ij}^k + f_{ij} y_{ij} \big) & \\ & \sum_{k\in K} d^k x_{ij}^k \leq u_{ij} y_{ij} & (i,j) \in A \\ & 0 \leq x_{ij}^k \leq u_{ij}^k y_{ij} & (i,j) \in A \ , \ k \in K \\ & y \in Y & \end{array}$$

- Still solvable if $Y \subset \{ 0 , 1 \}^{|A|}$ "not too nasty": first

$$\begin{array}{lll} f_{ij}^*(\pi) = \min & \sum_{k\in K} (c_{ij}^k - \pi_i^k + \pi_j^k) x_{ij}^k & \\ & \sum_{k\in K} d^k x_{ij}^k \leq u_{ij} & \\ & 0 \leq x_{ij}^k \leq u_{ij}^k & k \in K \end{array}$$

and then $\min \{ \sum_{(i,j)\in A} (f_{ij}^*(\pi) + f_{ij}) y_{ij} \ : \ y \in Y \}$

- Computational cost $\approx$ same but Lagrangian function no longer separable $\Longrightarrow$ wave goodbye to disaggregate master problem, easy components
- Still, the Lagrangian problem is somewhat separable
- We want to "show this quasi-separability to the master problem"

General setting: quasi-separable problems

- Set of N quasi-continuous (vector) variables x_i governed by y_i

$$\max\ dy + \sum_{i \in N} c_i x_i \tag{18}$$
$$Dy + \sum_{i \in N} C_i x_i = b \tag{19}$$
$$A_i x_i \leq b_i y_i \qquad i \in N \tag{20}$$
$$x_i \in X_i \qquad i \in N \tag{21}$$
$$y \in Y \tag{22}$$

- m linking constraints (19): Lagrangian relaxation

$$\phi(\lambda) = \lambda b + \max\ \left\{ (d - \lambda D)y + \sum_{i \in N}(c_i - \lambda C_i)x_i \ :\ (20)\ ,\ (21)\ ,\ (22) \right\}$$

- Two-stage solution procedure

$$\phi_i(\lambda) = \max\ \left\{ (c_i - \lambda C_i)x_i \ :\ x_i \in X_i \right\} \qquad i \in N \tag{23}$$

$$\phi(\lambda) = \lambda b + \max\ \left\{ \sum_{i \in N}(d_i - \lambda D^i + \phi_i(\lambda))y_i \ :\ y \in Y \right\} \tag{24}$$

Making it separable: the dumb way

- D-W reformulation is not disaggregate

$$\max \sum_{(\bar{y},\bar{x})\in YX} \left(d\bar{y} + \sum_{i\in N} c_i\bar{x}_i \right)\theta_{(\bar{y},\bar{x})} \tag{25}$$

$$\sum_{(\bar{y},\bar{x})\in YX} \left(D\bar{y} + \sum_{i\in N} C_i\bar{x}_i \right)\theta_{(\bar{y},\bar{x})} = b \tag{26}$$

$$\sum_{(\bar{y},\bar{x})\in YX} \theta_{(\bar{y},\bar{x})} = 1 \quad , \quad \theta_{(\bar{y},\bar{x})} \geq 0 \qquad (\bar{y},\bar{x}) \in YX \tag{27}$$

- Can be made so the hard way: also relax (20) ($\mu = [\mu_i]_{i\in N} \geq 0$)

$$\phi(\lambda,\mu) = \lambda b + \psi(\lambda,\mu) + \sum_{i\in N} \psi_i(\lambda,\mu_i) \qquad \text{with} \tag{28}$$

$$\psi_i(\lambda,\mu_i) = \max \left\{ (c_i - \lambda C_i - \mu_i A_i)x_i \ : \ x_i \in X_i \right\} \tag{29}$$

$$\psi(\lambda,\mu) = \max \left\{ \sum_{i\in N}(d_i - \lambda D^i - \mu_i b_i)y_i \ : \ y \in Y \right\} \tag{30}$$

- Many more multiplayers ($|K||A|$ in FC-MMCF)
- Can easily destroy any advantage due to separability

Making it separable: the better way

- "Easy component" Y version:

$$\max\ dy + \sum_{i \in N} \sum_{\bar{x}_i \in X_i} (c_i \bar{x}_i)\theta_{\bar{x}_i} \tag{31}$$

$$Dy + \sum_{i \in N} \sum_{\bar{x}_i \in X_i} (C_i \bar{x}_i)\theta_{\bar{x}_i} = b \tag{32}$$

$$\sum_{\bar{x}_i \in X_i} (A_i \bar{x}_i)\theta_{\bar{x}_i} \leq y_i \qquad i \in N \tag{33}$$

$$\sum_{\bar{x}_i \in X_i} \theta_{\bar{x}_i} = 1 \qquad i \in N \tag{34}$$

$$y \in Y \ , \ \theta_{\bar{x}_i} \geq 0 \qquad \bar{x}_i \in X_i \ , \ i \in N$$

- Nifty idea: replace (33)–(34) with

$$\sum_{\bar{x}_i \in \bar{X}_i} \theta_{\bar{x}_i} = y_i \qquad i \in N \tag{35}$$

  then relax (35) with multipliers $\gamma = [\gamma_i]_{i \in N} \geq 0$

- Multipliers are from master problem constraints (which they are . . .)
- Non-easy component version obvious
- Much fewer multipliers (1 instead of m), much more elegant

And it also works in practice

- Results from last week (Enrico is the pit bull of numerical experiments)
- Time limit 18000 seconds (always hit if not shown)

	BKA-10	BKA-4000		BKD	BQS	
name	gap	time	gap	gap	time	gap
p33	5.71e-06	227.68	6.58e-07	4.63e-02	5.27	1.31e-07
p34	8.20e-06	233.14	3.47e-07	5.43e-02	5.36	3.31e-07
p35	7.33e-06	260.01	8.63e-07	8.92e-02	5.83	3.27e-09
p36	9.61e-06	57.02	8.48e-07	9.33e-02	4.59	3.85e-07
p37	5.14e-04	—	3.22e-04	9.23e-02	3954.59	1.44e-07
p38	4.79e-04	—	3.24e-04	5.75e-02	3724.92	2.58e-07
p39	4.54e-06	—	2.46e-05	4.46e-02	964.00	1.33e-09
p40	4.99e-06	—	1.45e-05	5.13e-02	838.73	4.71e-09
p41	3.22e-06	212.67	3.13e-08	4.92e-02	6.75	2.54e-08
p42	3.29e-06	130.07	2.58e-08	7.34e-02	6.66	2.79e-10
p43	9.91e-06	193.61	2.97e-08	8.99e-02	5.25	5.89e-10
p44	5.16e-06	134.04	1.28e-06	1.34e-01	6.56	2.34e-07

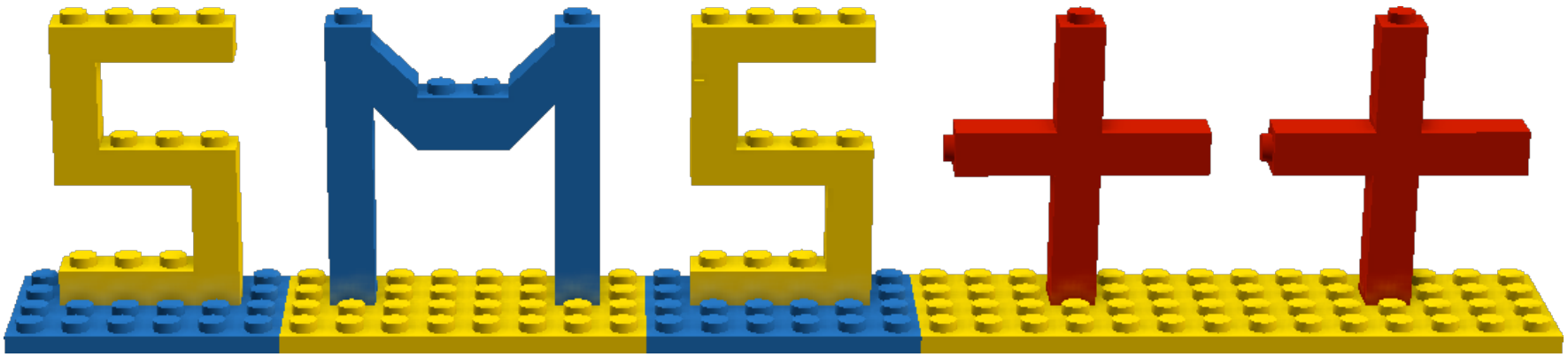
- Our last paper all together[36]

[36] F., Gendron, Gorgone "Separable Lagrangian Decomposition for Quasi-Separable Problems" *Bernard's Book*, 2023

But Bernard's legacy will live on, also in software

- Putting these ideas in practice: easier said than done
- Specialized implementations for one application "relatively easy"
- General implementations for all problems with same structure harder:
  it took $\approx$ 10 years from idea to paper for easy components
  on top of existing, nicely structured C++ bundle code
- It's 10 years since S^2DW and we still don't have a general implementation
- Issue: extracting structure from problems
- Issue: really using this in a B&C approach
  $\approx$ 20 years doing this well for Multicommodity Network Design
- Especially hard: multiple nested forms of structure, reformulation
- Current modelling/solving tools just don't do it
- So I have been building my own

Meet SMS++



https://gitlab.com/smspp/smspp-project

"For algorithm developers, from algorithm developers"

- Open source (LGPL3)
- 1 "core" repo, 1 "umbrella" repo, 10+ problem and/or algorithmic-specific repos (public, more in development)
- Extensive Doxygen documentation https://smspp.gitlab.io
- But no real user manual as yet

What SMS++ is

- A core set of C++-17 classes implementing a modelling system that:
  - explicitly supports the notion of Block ≡ nested structure
  - separately provides "semantic" information from "syntactic" details (list of constraints/variables ≡ one specific formulation among many)
  - allows exploiting specialised Solver on Block with specific structure
  - manages any dynamic change in the Block beyond "just" generation of constraints/variables
  - supports reformulation/restriction/relaxation of Block
  - has built-in parallel processing capabilities
  - should be able to deal with almost anything (bilevel, PDE, ...)
- An hopefully growing set of specialized Block and Solver
- In perspective an ecosystem fostering collaboration and code sharing: a community-building effort as much as a (suite of) software product(s)
- I believe Bernard would have loved it

And finally the really important things

And finally the really important things

And finally the really important things

And finally the really important things

And finally the really important things