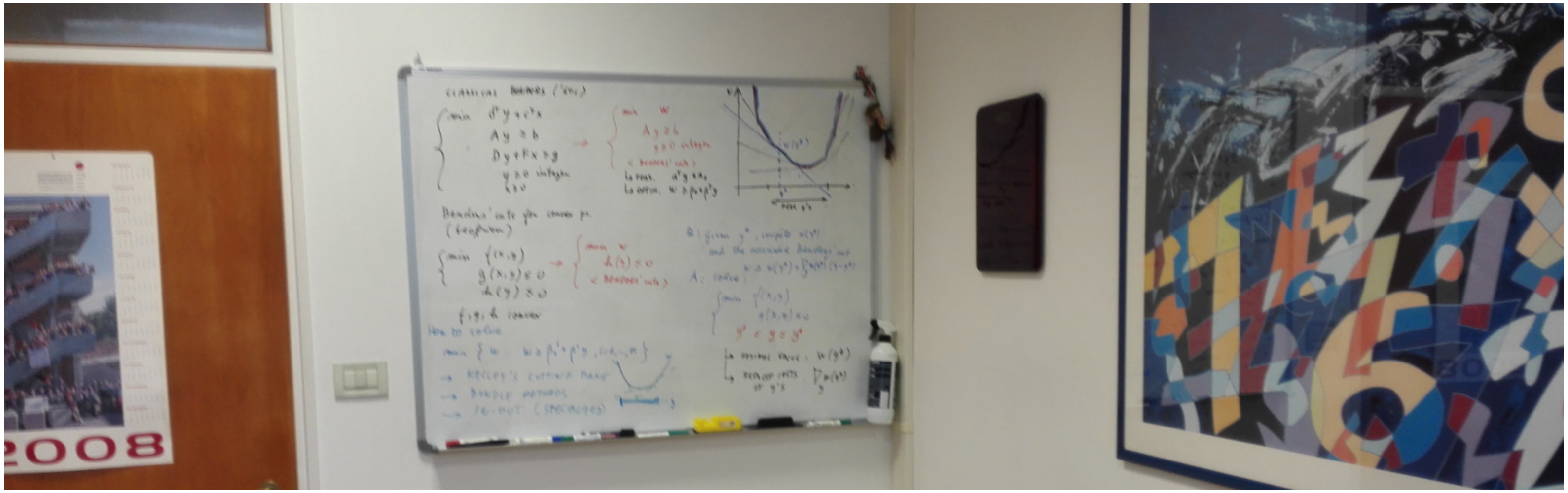
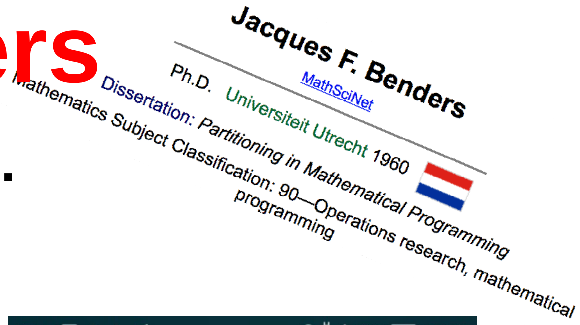
Thinning out facilities: Lagrange, Benders, and (the curse of) Kelley

Matteo Fischetti, University of Padova

Markus Sinnl, Ivana Ljubic, University of Vienna



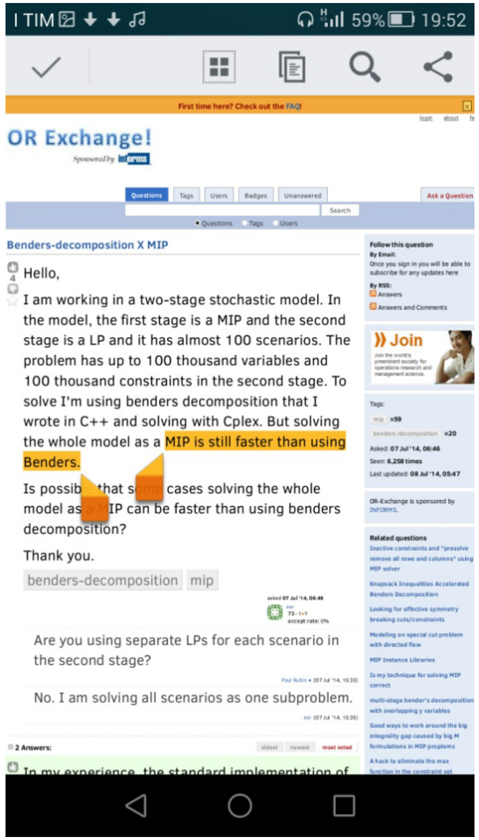
Apology of Benders



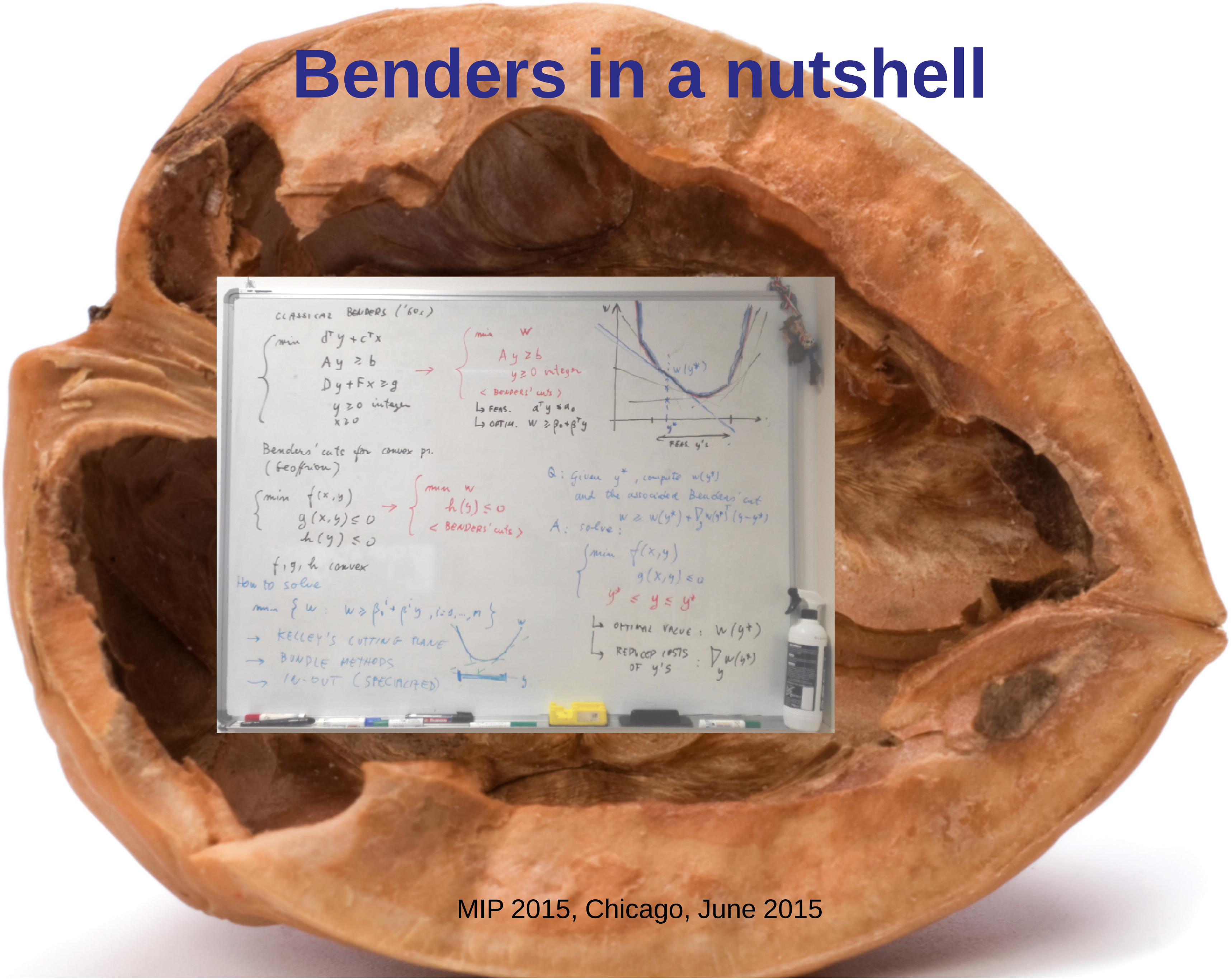
Everybody talks about Benders decomposition…

… but not so many MIPeople actually use it

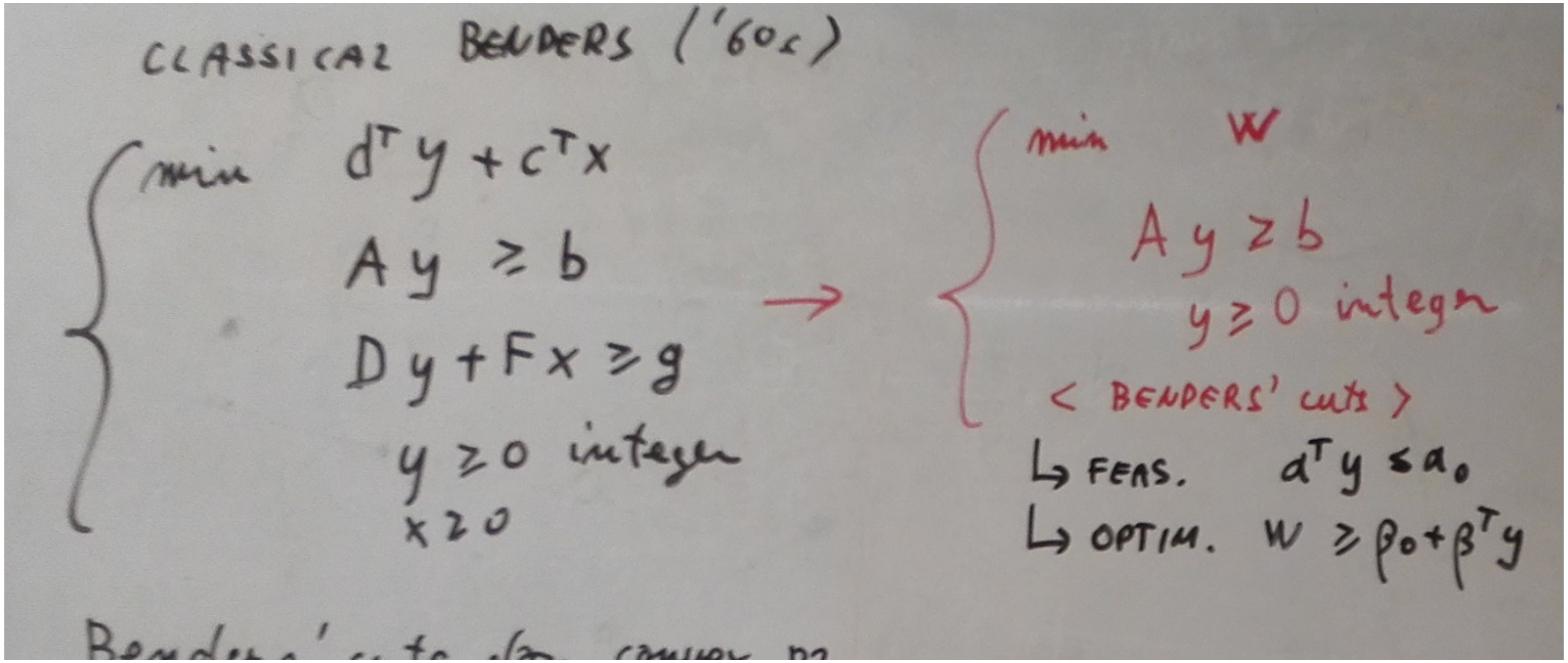
… besides Stochastic Programming guys of course



Benders in a nutshell



#BendersToTheBone



Original problem (left) vs Benders' master problem (right)

Benders after Padberg&Rinaldi

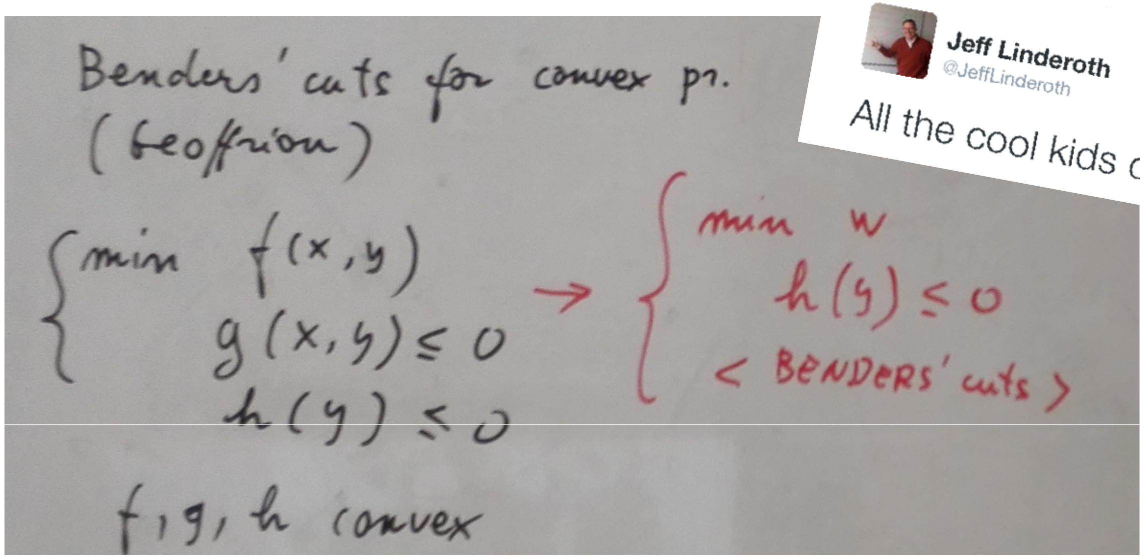
- The original ('60s) recipe was to solve the master to optimality by enumeration (integer y*), to generate B-cuts for y*, and to repeat

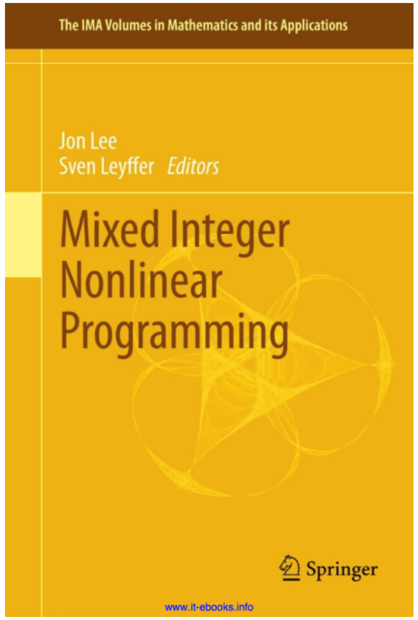
  → This is what we call "**Old Benders**" within our group

  → **still the best option for some problems!**
- Folklore (Miliotios for TSP?): generate B-cuts for any integer y* that is going to update the incumbent
- McDaniel & Devine (1977) use of B-cuts to cut (root node) fractional y*'s
- …
- Everything fits very naturally within modern **Branch-and-Cut**
  - **Lazy constraint** callback for integer y* (needed for correctness)
  - **User cut** callback for any y* (useful but not mandatory)

- Feasibility cuts → we know how to handle (minimal infeasibility etc.)
- **Optimality cuts → often a nightmare even after MW improvements (pareto-optimality) and alike → THE TOPIC OF THE PRESENT TALK**

Benders for convex MINLP

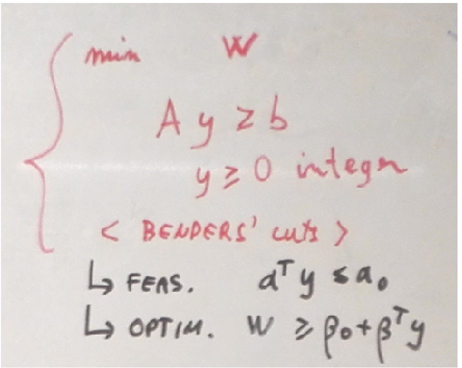


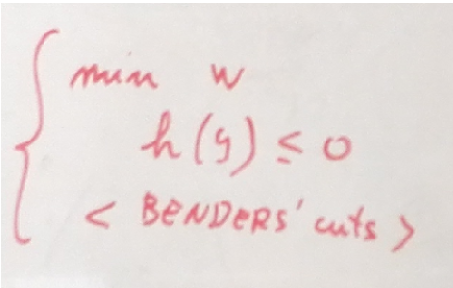


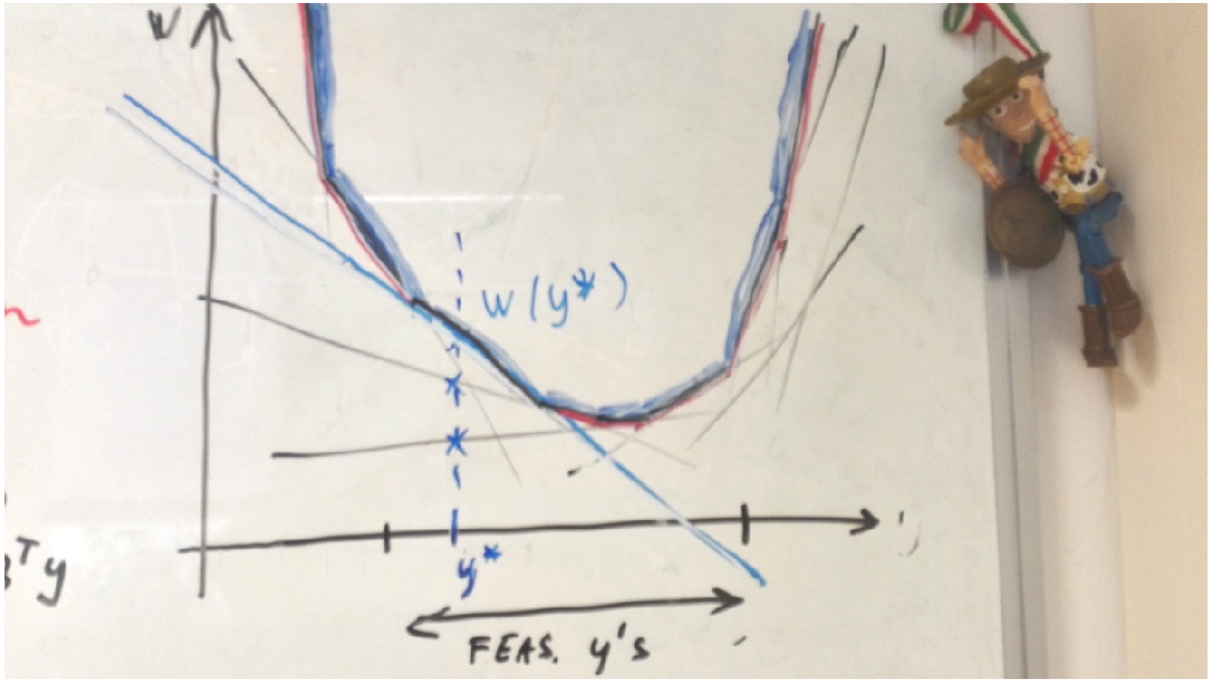
- Benders cuts can be generalized to convex MINLP
  - → Geoffrion via Lagrangian duality
  - → resulting **Generalized Benders** cuts still linear

- Potentially very useful **to remove nonlinearity** from the master by using kind of “surrogate cone” cuts → hide nonlinearity where it does not hurt…

Optimality cut geometry

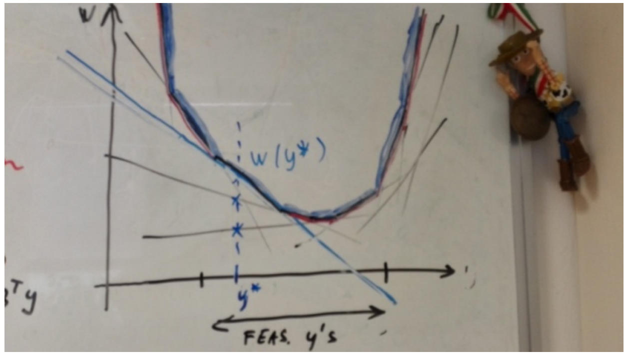




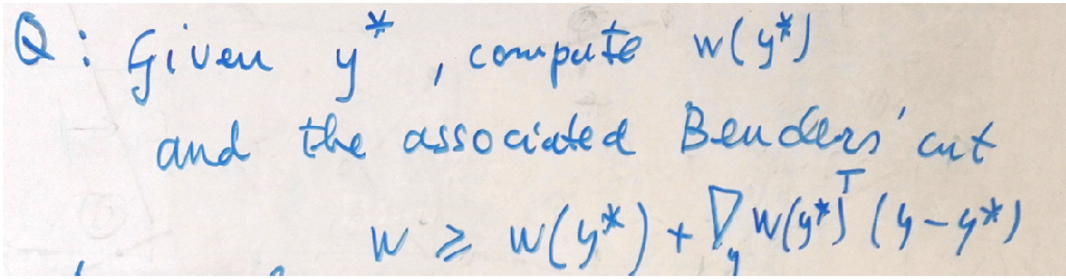


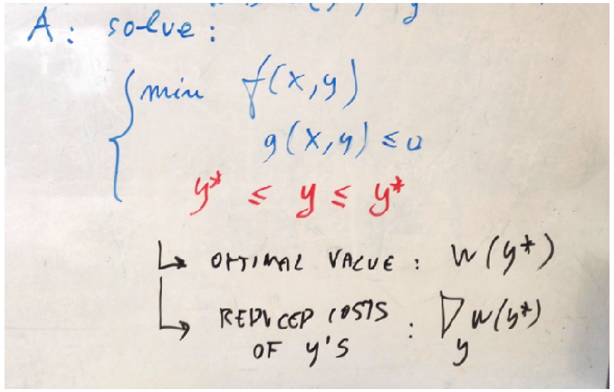
Solving the master LP relaxation → minimization of a convex function w(y) → a very familiar setting for people working with **Lagrange** duality (Dantzig-Wolfe decomposition and alike)   **#LagrangeEverywhere**

Optimality cut generation



Given y*, how to compute the supporting hyperplane (in blue)?





1-2-3 Benders optimality cut computation

1) solve the original convex problem with new var. bounds $y^* \leq y \leq y^*$

2) take *opt_val* and reduced costs r_j's

3) write $w \geq opt\_val + \sum_j r_j (y_j - y_j^*)$

Benders++ cuts

- We have seen that Benders cuts are obtained by solving the **original problem** after fixing y=y*, thus voiding the information that y must be integer

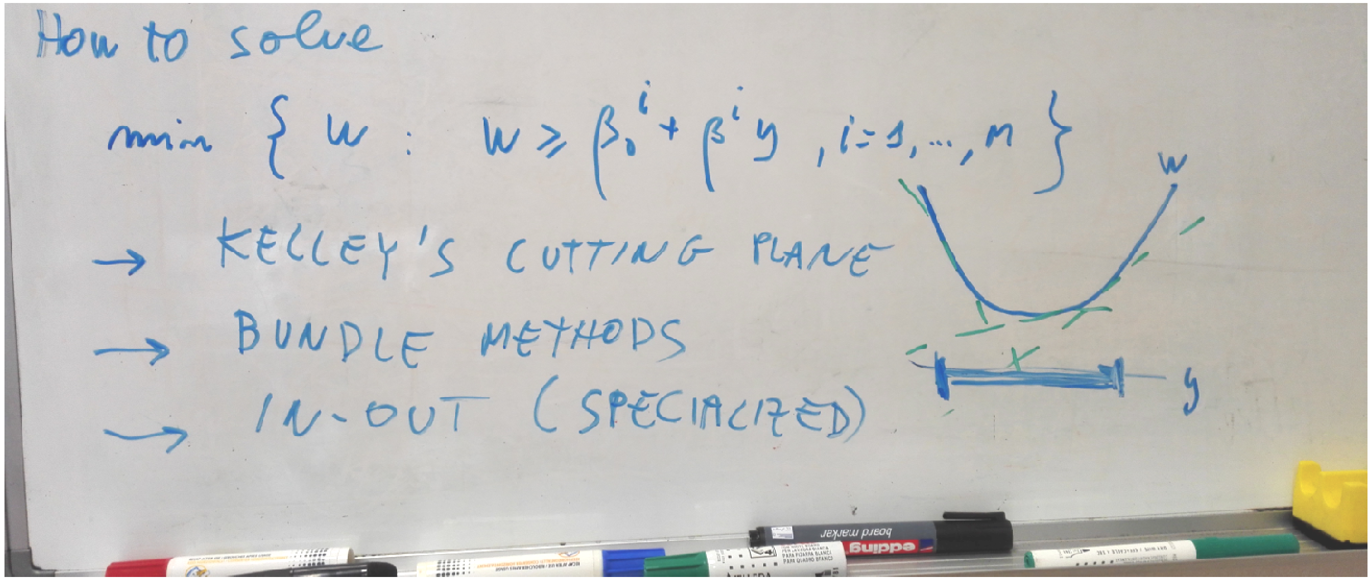
- Full primal optimal sol. **(y*,x*)** available for generating MIP cuts exploiting the integrality of y

- However **(y*,x*)** is not a vertex → no cheap "tableau cuts" (GMI and alike) available …
 … while any black-box **separation function** that receives the original model and the pair (y*,x*) on input can be used (MIR heuristics, CGLP's, half cuts, etc.)

- Generated cuts to be added to the original model (i.e. to the "slave") in case they involve the x's

- Very good results with split cuts for Stochastic Integer Programming recently reported by Bodur, Dash, Gunluck, Luedtke (2014)

#TheCurseOfKelley



Now that you have seen the plot of **w(y)**, you understand a main reason for Benders slow convergence → if still skeptical, please call one of these guys…



UFL with linear and quadratic costs

- **Uncapacitated Facility Location** (a.k.a. Simple Plant Location in the old days…)
- One of the basic OR problems, deeply studied in the 70-80' by pioneers like Balas, Geoffrion, Magnanti, Cornuejols, Nemhauser, Wolsey, …

$$\min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$\begin{aligned}
\text{s.t.} \ \sum_{i \in I} x_{ij} &= 1 && \forall j \in J \\
x_{ij} &\le y_i && \forall i \in I, j \in J \\
x_{ij} &\ge 0 && \forall i \in I, j \in J \\
y_i &\in \{0,1\} && \forall i \in I
\end{aligned}$$

UFL (linear costs) MIP model

$$
\begin{aligned}
\min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} & & \\
\text{s.t.} \ \sum_{i \in I} x_{ij} = 1 & & \forall j \in J \\
x_{ij} \le y_i & & \forall i \in I, j \in J \\
x_{ij} \ge 0 & & \forall i \in I, j \in J \\
y_i \in \{0,1\} & & \forall i \in I
\end{aligned}
$$

- Can be viewed as a **2-stage Stochastic Program**: pay to open facilities in the first stage, get a second-stage cost correction by each client (scenario) → x's are just "recourse var.s"

- **Benders decomposition**: very natural, potentially very useful, addressed in the early days but apparently dismissed nowadays

- **Current best exact solver**: Lagrangian optimization (Posta, Ferland, Michelon, 2014)

qUFL (quadratic costs)

- Just change objective to $\min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}^2$
- Applications in energy systems with power losses (dispersion → electrical currents' square) and finance applications (variance)
- Embarrassingly tight **perspective** reform. (Gunluk, Linderoth, 2012)

$$
\begin{aligned}
\min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} z_{ij} & & \\
\text{s.t.} \sum_{i \in I} x_{ij} = 1 & & \forall j \in J \\
x_{ij} \le y_i & & \forall i \in I, j \in J \\
x_{ij}^2 \le z_{ij}\, y_i & & \forall i \in I, j \in J \\
x_{ij} \ge 0 & & \forall i \in I, j \in J \\
z_{ij} \ge 0 & & \forall i \in I, j \in J \\
y_i \in \{0, 1\} & & \forall i \in I
\end{aligned}
$$

Our specialized Benders

- **Fat** master model: $\min \sum_{i \in I} f_i y_i + \sum_{j \in J} w_j$
- **Slim** (aggregated) master: $w_{sum} = \sum_{j \in J} w_j$
- **Specialized** slave solver (LP/QCP) for Benders cut generation:
  - faster
  - numerically more accurate

Algorithm 1: Our specialized QP solver

Input : A point $y^* > 0$ with $\sum_{i \in I} y_i^* \geq 2$ along with the cost vector $\gamma > 0$

Output: Optimal value vopt, optimal primal solution $x^* \geq 0$, and optimal dual solution $(u^*, \beta^*) \geq 0$ for the quadratic model (30)-(32)

* compute optimal primal solution x^* along with β^* */

$R \leftarrow I$;

$r \leftarrow 1$;

again ← TRUE;

while ($r > 0$ and again) **do**

  $\beta^* \leftarrow 2r / \sum_{i \in R}(1/\gamma_i)$;

  again ← FALSE ;

  foreach $i \in R$ **do**

    $x_i^* \leftarrow \beta^* \,/\, (2\gamma_i)$;

    if ($x_i^* > y_i^*$) **then**

      $x_i^* \leftarrow y_i^*$;

      $r \leftarrow r - y_i^*$;

      $R \leftarrow R \setminus \{i\}$;

      again ← TRUE;

    end

  end

end

/* compute optimal value vopt and dual solution u^* */

if ($r \leq 0$ or $R = \emptyset$) **then** $\beta^* \leftarrow 0$;

vopt ← 0;

for $i \in I$ **do**

  if $i \in R$ **then** $u_i^* \leftarrow 0$ **else** $u_i^* \leftarrow \beta^* - 2\gamma_i y_i^*$;

  vopt ← vopt $+ \gamma_i (x_i^*)^2$;

end

- **Specialized** UFL heuristic (linear case only)
- Margot's test of cut validity (very useful to trap numerical troubles)

Escaping the #CurseOfKelley

- Root node LP bound **very critical** → many ships sank here!

- **Kelley's** cutting plane can be desperately slow, bundle methods required

- In a root node preprocessing, we implemented our own "interior point" method inspired by (Ben-Ameur and Neto 2007, Fischetti and Salvagnin 2010, Naoum-Sawaya and Elhedhli 2013).

- Note that every point y in the 0-1 hypercube is "internal" to the (y,w) polyhedron for a sufficiently large w → **you better work on the y-space** (as any honest bundle would do)

- In-out/analytic center methods work on the (y,w) space → adaptation needed

- As a quick shot, we implemented a very simple **"chase the carrot"** heuristic to determine an internal path towards the optimal y

- Our very first implementation worked so well that we did not have an incentive to try and improve it **#OccamPrinciple**

Our #ChaseTheCarrot dual heuristic

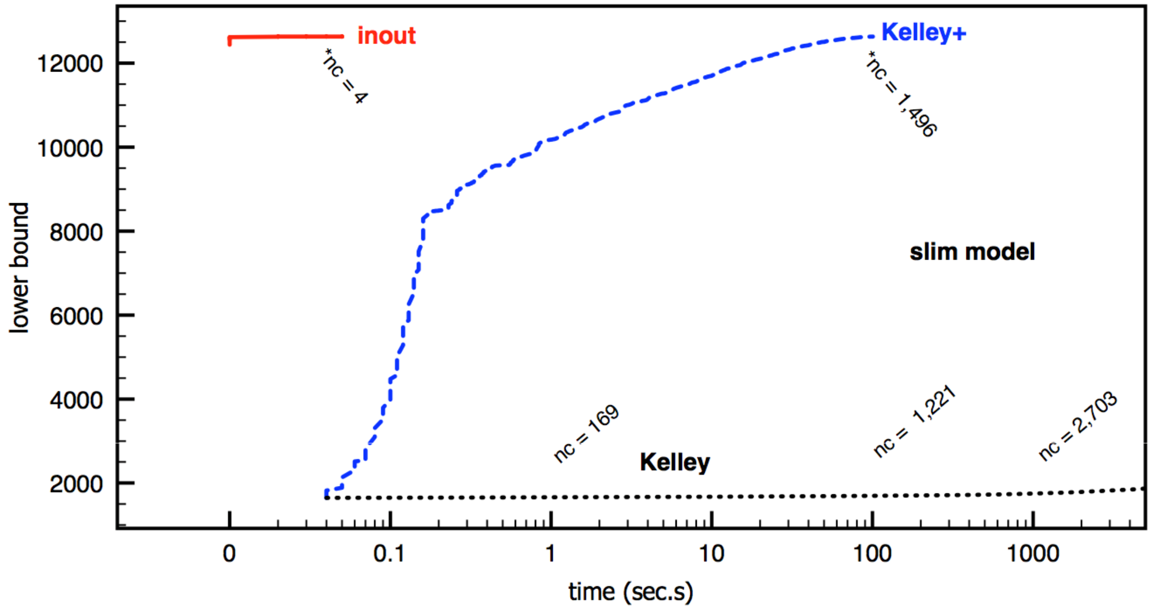
• We (the donkey) start with y=(1,1,…) and optimize the master LP as in Kelley, to get optimal y* (the carrot on the stick).

• We move y half-way towards y*. We then separate a point y' in the segment y-y* close to y. The generated optimality cut(s) are added to the master LP, which is reoptimzied to get the new optimal y* (carrot moves).

• Repeat until bound improves, then switch to Kelley for final bound refinement (cross-over like)

• Warning: adaptations needed if feasibility cuts can be generated…

Effect of the improved cut-loop



- Comparing **Kelley** cut loop at the root node with **Kelley+** (add epsilon to y*) and with our chase-the-carrot method (**inout**)
- Koerkel-Ghosh **qUFL** instance gs250a-1 (250x250, quadratic costs)
- ***nc** = n. of Benders cuts generated at the end of the root node
- times in **logarithmic scale**

Computational results (linear case)

- Many hard instances from UFLLIB solved in just sec.s
- Some instances solved to proven optimality for the first time

Table 1 Previously unsolved UFL instances solved to optimality using our approach (linear costs).

inst.	bestknown	opt	$t[s]$	rootbound	$t_{root}[s]$	$g_r[\%]$	nodes
ga250a-3	257985	257953	493.49	257554.773407	12.77	0.15	200184
ga250a-5	258225	258190	585.93	257790.245068	9.65	0.15	229446
ga500c-5	621313	621313	9226.86	601500.282332	12.31	3.19	195191
gs500c-3	621204	621204	11448.19	601980.526816	13.44	3.09	194657
gs500c-5	623180	623180	26828.91	603115.401650	14.20	3.22	270147
2500-10	3101800	3099907	824.76	3097480.189279	104.67	0.08	1362
3000-100	1602335	1602154	225.25	1601733.816607	82.67	0.03	441

- Many best-known solution values strictly improved (22 out of 50) or matched (22 more).

Computational results (quadratic case)

Table 3 **Comparing our slim and fat models with the perspective reformulation (Günlük and Linderoth 2012), on a set of randomly generated qUFL instances proposed in Günlük et al. (2007), Günlük and Linderoth (2012). Perspective reformulation hits memory limit for $n, m \geq 200$.**

		Our slim model				Our fat model				Perspective reformulation			
n	m	$t[s]$	$g_r[\%]$	$t_{root}[s]$	nodes	$t[s]$	$g_r[\%]$	$t_{root}[s]$	nodes	$t[s]$	$g_r[\%]$	$t_{root}[s]$	nodes
50	50	0.04	0.14	0.03	1.6	0.05	0.13	0.03	1.6	39.89	0.07	29.08	4.2
50	100	0.06	0.13	0.04	2.6	0.10	0.11	0.06	2.7	105.77	0.14	63.18	7.6
50	200	0.11	0.13	0.08	6.7	0.29	0.12	0.16	7.5	195.47	0.13	90.43	10.0
80	30	0.05	0.22	0.03	1.7	0.05	0.16	0.03	1.1	72.42	0.29	41.44	6.0
80	50	0.08	0.36	0.05	5.7	0.07	0.34	0.04	5.5	137.80	0.40	61.77	10.9
80	100	0.10	0.21	0.08	5.9	0.14	0.21	0.08	5.3	279.94	0.25	120.10	8.0
80	200	0.14	0.13	0.11	5.2	0.27	0.14	0.16	6.4	622.38	0.15	202.79	11.5
100	100	0.23	0.21	0.19	6.6	0.16	0.20	0.10	6.0	563.33	0.25	208.60	13.0
150	150	0.24	0.17	0.19	7.8	0.32	0.16	0.20	9.0	2526.73	0.17	869.19	11.9
200	200	0.33	0.06	0.28	6.7	0.45	0.06	0.32	4.1	—	—	—	—
250	250	0.46	0.05	0.42	4.3	0.71	0.04	0.60	4.1	—	—	—	—

Up to **10,000 speedup** for medium-size instances (150x150)

Much larger instances (250x250) solved in less than 1 sec.

Computational results (quadratic case)

Table 4 Comparing the performance of slim versus fat model on a larger set of benchmark instances for qUFL generated as in Günlük et al. (2007), Günlük and Linderoth (2012).

		Our slim model				Our fat model			
n	m	$t[s]$	$g_r[\%]$	$t_{root}[s]$	nodes	$t[s]$	$g_r[\%]$	$t_{root}[s]$	nodes
500	500	1.39	0.03	1.31	16.2	3.30	0.03	2.82	9.5
500	1000	3.02	0.03	2.75	54.7	8.90	0.03	7.81	20.8
500	5000	11.59	0.01	10.41	87.2	132.89	0.02	127.27	32.4
500	10000	36.98	0.01	22.09	558.2	673.93	0.02	646.97	106.5
1000	500	3.80	0.04	3.32	76.0	4.60	0.04	3.86	26.1
1000	1000	5.78	0.03	5.25	65.3	15.18	0.03	13.74	28.2
1000	5000	20.70	0.01	19.32	44.3	193.76	0.02	181.87	180.3
1000	10000	64.01	0.01	34.74	603.0	799.02	0.02	748.56	399.8
2000	500	6.73	0.03	6.10	66.7	8.95	0.03	7.83	29.8
2000	1000	14.86	0.02	12.72	194.4	35.41	0.02	32.65	65.9
2000	5000	115.09	0.01	42.07	1649.0	405.85	0.02	361.69	629.3
2000	10000	309.36	0.01	76.88	10735.8	2646.69	0.03	1246.60	13114.0

Huge instances (2,000x10,000) solved in 5 minutes

MIQCP's with 20M SOC constraints and 40M var.s

qUFL much easier than UFL (!)

Table 5 All KG instances for qUFL are solved to optimality by our slim model (quadratic costs). Each row shows average values over 5 instances per subclass.

group	$t[s]$	$g_r[\%]$	$t_{root}[s]$	nodes
ga250a	0.40	0.00	0.39	4.4
ga250b	0.28	0.03	0.22	71.2
ga250c	0.40	0.03	0.36	39.4
gs250a	0.21	0.00	0.20	3.4
gs250b	0.27	0.02	0.21	80.6
gs250c	0.46	0.03	0.42	21.6
ga500a	0.76	0.00	0.73	3.0
ga500b	2.12	0.04	1.95	58.0
ga500c	19.46	0.16	1.49	49911.6
gs500a	0.81	0.00	0.77	12.4
gs500b	2.47	0.03	2.31	72.8
gs500c	15.05	0.14	1.26	12721.6
ga750a	2.03	0.00	1.62	107.4
ga750b	2.08	0.01	1.82	65.2
ga750c	35.79	0.08	2.41	64338.0
gs750a	1.94	0.00	1.65	53.2
gs750b	3.24	0.01	1.82	414.0
gs750c	26.94	0.07	2.98	16837.0

- Due to the extremely tight lower bound, the quadratic case is typically **orders of magnitude easier** than its linear counterpart!

- Of course only when **Benders** is used to control
  - n. of **variables**
  - n. of **SOC constraints**

  and to hide nonlinearity where it **does not hurt** (in the slave) while the master remains a neat MILP

Thanks for your attention

- Full paper

  M. Fischetti, I. Ljubic, M. Sinnl, "Thinning out facilities: a Benders decomposition approach for the uncapacitated facility location problem with separable convex costs", Tech. Rep. UniPD, 2015.

  and slides available at

  http://www.dei.unipd.it/~fisch/papers/
  http://www.dei.unipd.it/~fisch/papers/slides/

- Thanks are due to @Fischeders who was supposed to deliver this talk but did not show up on time #TooNerd

Some references

Ben-Ameur, W., J. Neto. 2007. Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks* **49** 3–17.

Bonami, P., M. Kilinc, J. Linderoth. 2012. Algorithms and software for convex mixed integer nonlinear programs. *Mixed Integer Nonlinear Programming* **154** 1–39.

Cornuejols, G., G.L. Nemhauser, L.A. Wolsey. 1980. A canonical representation of simple plant location problems and its applications. *SIAM J. Algebr. Discr. Meth.* **1** 261–272.

Fischetti, M., D. Salvagnin. 2010. An in-out approach to disjunctive optimization. A. Lodi, M. Milano, P. Toth, eds., *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, *Lecture Notes in Computer Science*, vol. 6140. Springer Berlin Heidelberg, 136–140.

Frangioni, A., C. Gentile. 2006. Perspective cuts for a class of convex 0-1 mixed integer programs. *Math. Programming* **106** 225–236.

Geoffrion, A. 1972. Generalized Benders Decomposition. *J. Optim. Theory Appl.* **10** 237–260.

Some references

Günlük, O., J. Lee, R. Weismantel. 2007. MINLP strenghtening for separable convex quadratic transportation-cost UFL. Tech. Rep. RC24213 (W0703-042), IBM Research Division.

Günlük, O., J. Linderoth. 2012. Perspective reformulation and applications. J. Lee, S. Leyffer, eds., *Mixed Integer Nonlinear Programming*. Springer, 61–92.

Hijazi, H., P. Bonami, A. Ouorou. 2014. An outer-inner approximation for separable mixed-integer nonlinear programs. *INFORMS J. Comput.* **26** 31–44.

Naoum-Sawaya, Joe, Samir Elhedhli. 2013. An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Ann. Oper. Res.* **210** 33–55.

Posta, M., J. A. Ferland, P. Michelon. 2014. An exact cooperative method for the uncapacitated facility location problem. *Math. Programming Comput.* **6** 199–231.

and of course

Jacques F. Benders

MathSciNet

Ph.D. Universiteit Utrecht 1960

Dissertation: *Partitioning in Mathematical Programming*

Mathematics Subject Classification: 90—Operations research, mathematical programming