Simplified Benders cuts for Facility Location

Matteo Fischetti, University of Padova

based on joint work with
Ivana Ljubic (ESSEC, Paris) and Markus Sinnl (ISOR, Vienna)

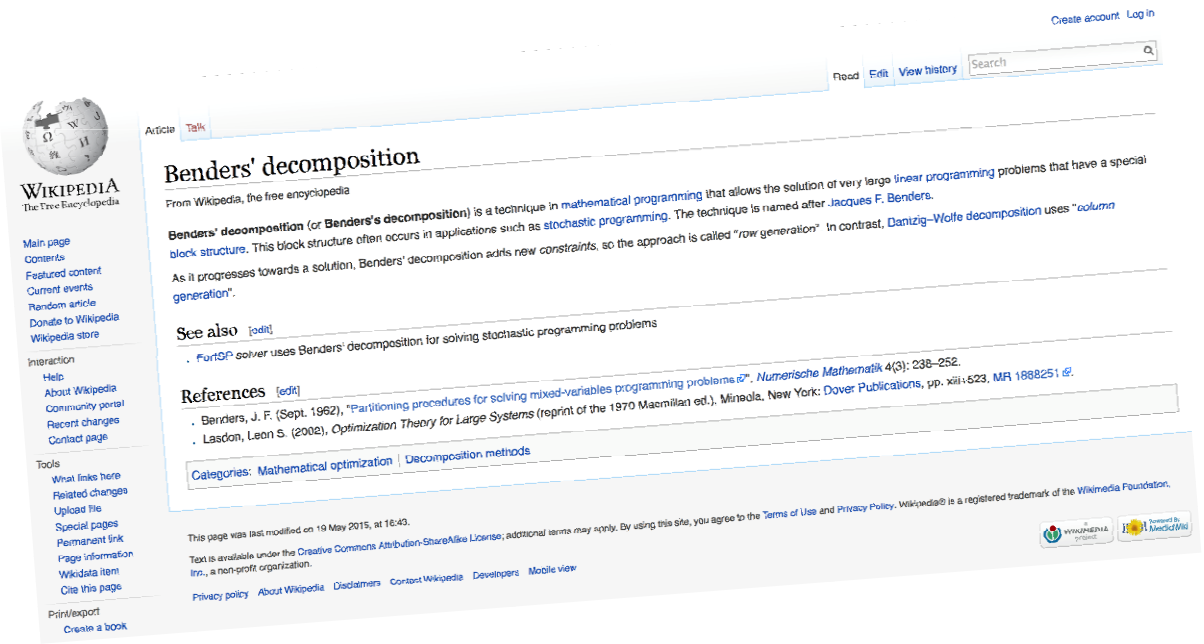
Apology of Benders

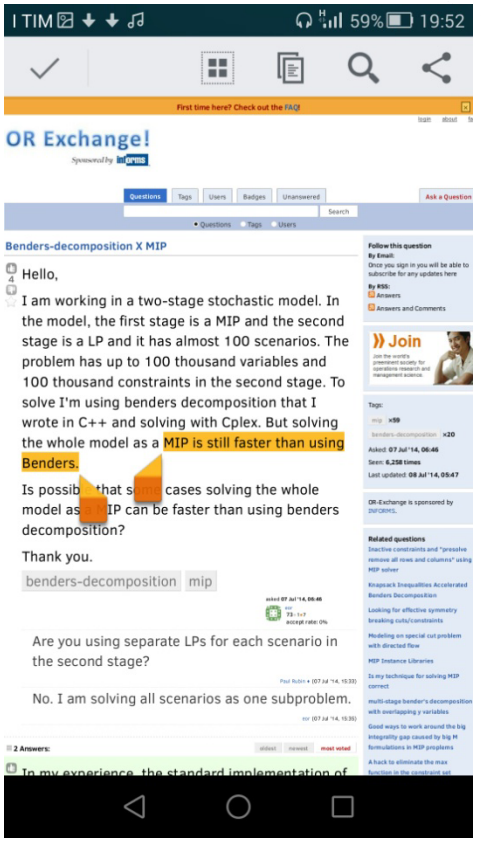
Jacques F. Benders
MathSciNet
Ph.D. Universiteit Utrecht 1960
Dissertation: Partitioning in Mathematical Programming
Mathematics Subject Classification: 90—Operations research, mathematical programming

Everybody talks about Benders decomposition…

… but not so many MIPeople actually use it

…because of its slow-convergence reputation…



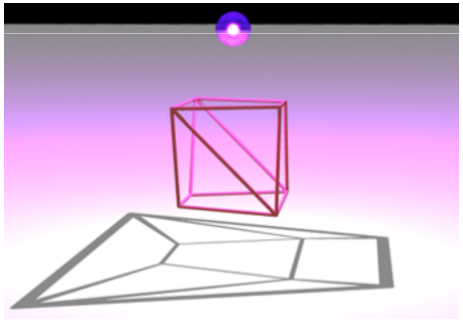


Benders: why and how

- Benders decomposition is aimed at solving an optimization problem living in the (x,y) space by working on the y subspace only (**master problem**)

- As such, it is just a **projection** tool whose application requires some assumptions (essentially, convexity on the x-space)

- Projection is achieved by means of cuts in the y-space → the (in)famous Benders' **feasibility** and **optimality** cuts obtained by solving a certain "slave subproblem"



- **Mixed** results reported in practice: **it can work very well or very bad**

Benders after Padberg & Rinaldi

- Typical application in MILP: y integer (master var.s), x continuous.

- 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 within a **single branching tree**
- McDaniel & Devine (1977) use of B-cuts to cut (root node) **fractional** y*’s

- Fits well within **modern Branch-and-Cut** **#JustAnotherFamilyOfCuts**
  - **Lazy constraint** callback for integer y* (needed for correctness)
  - **User cut** callback for any y* (useful but not mandatory)

A successful application: UFL

- **Uncapacitated Facility Location** (a.k.a. Simple Plant Location)

- One of the basic OR problems, deeply studied in the 70-80' by pioneers like Balas, Geoffrion, Magnanti, Cornuejols, Nemhauser, Wolsey, …

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

$$\text{s.t.} \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J$$

$$y_i \in \{0,1\} \qquad \forall i \in I$$

UFL (linear costs) MIP model

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

$$\text{s.t.} \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J$$

$$y_i \in \{0,1\} \qquad \forall i \in I$$

- 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 forgotten nowadays

- **Best exact solver** from literature: Lagrangian optimization (Posta, Ferland, Michelon, 2014)

Quadratic UFL (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)

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

$$\text{s.t.} \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J$$

$$x_{ij}^2 \leq z_{ij}\, y_i \qquad \forall i \in I, j \in J$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J$$

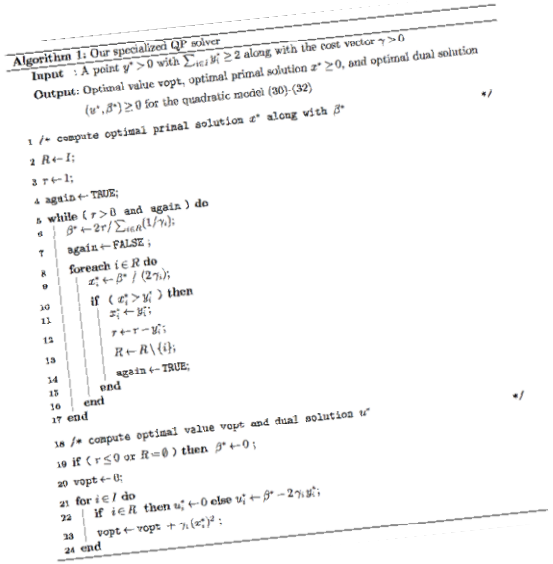
$$z_{ij} \geq 0 \qquad \forall i \in I, j \in J$$

$$y_i \in \{0, 1\} \qquad \forall i \in I$$

An effective branch-and-cut code

- Benders cuts embedded within Cplex's B&C through **callbacks**

- **Specialized** slave solver (LP/QCP) for Benders cut generation:
  - faster
  - numerically more accurate



Algorithm 1: Our specialized QP solver

Input : A point $y^* \geq 0$ 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 ← I;
r ← 1;
again ← TRUE;
while ( r > 0 and again ) do
    β* ← 2r / Σ_{i∈R} (1/γ_i);
    again ← FALSE ;
    foreach i ∈ R do
        x*_i ← β* / (2γ_i);
        if ( x*_i > y*_i ) then
            x*_i ← y*_i ;
            r ← r − y*_i ;
            R ← R \ {i};
            again ← TRUE;
        end
    end
end
/* compute optimal value vopt and dual solution u* */
if ( r ≤ 0 or R = ∅ ) then β* ← 0 ;
vopt ← 0;
for i ∈ I do
    if i ∈ R then u*_i ← 0 else u*_i ← β* − 2γ_i y*_i ;
    vopt ← vopt + γ_i (x*_i)^2 ;
end
```

- A basic version of this code is just a homework assignment for my students in Padua (computer science engineers)

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

Capacitated Facility Location

Each facility can support only a limited set of customers (capacity constraint)

$$\min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} d_i c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I$$

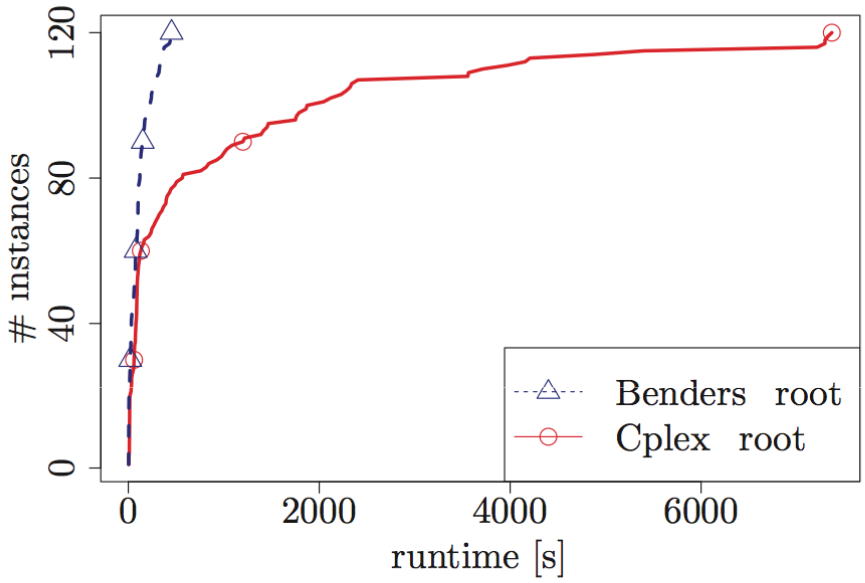
$$x_{ij} \leq y_j \qquad \forall i \in I, j \in J$$

$$\sum_{i \in I} d_i x_{ij} \leq s_j y_j \qquad \forall j \in J$$

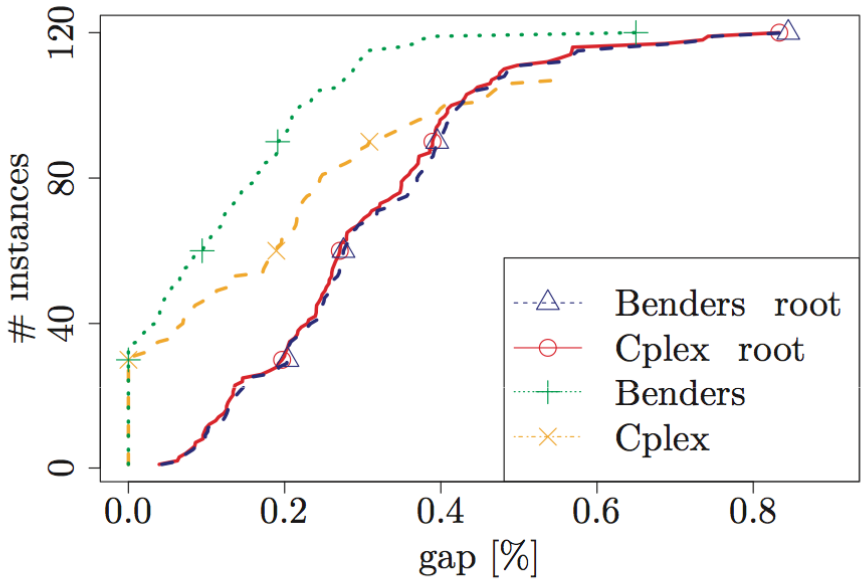
$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J$$

$$y_j \in \{0, 1\} \qquad \forall j \in J$$

Computational tapas



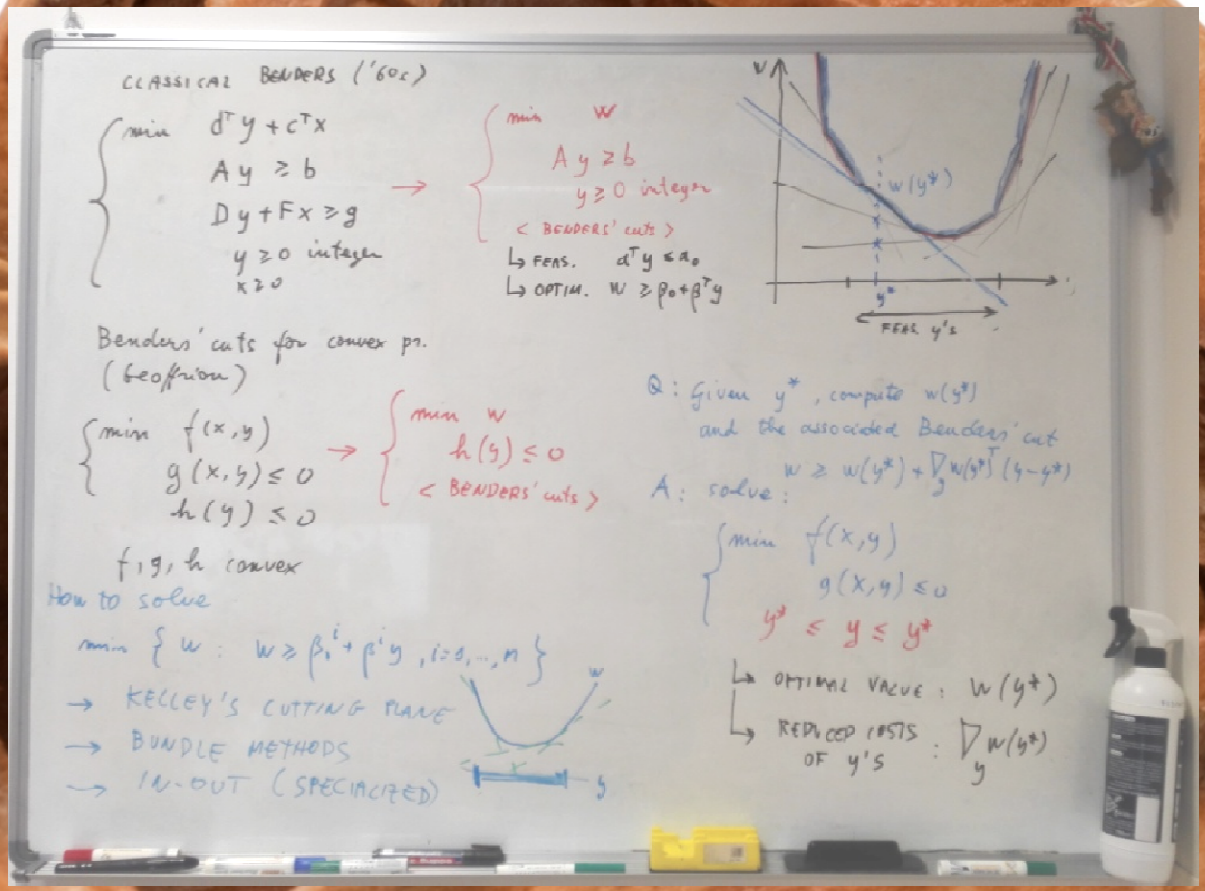
(a) Computing times at the root node.



(b) Relative gaps to the best known upper bounds.

Figure 1: GK instances: Comparing the performance of Benders and IBM ILOG Cplex 12.6.1 at the root node, and after a timelimit of 3600 seconds.

Benders in a nutshell



Modern Benders

Consider the original **convex** MINLP

$$\begin{aligned} \min\ & f(x,y) \\ & g(x,y) \leq 0 \\ & Ay \leq b \\ & y \text{ integer} \end{aligned}$$

and assume for the sake of simplicity

$$S := \{y : Ay \leq b\} \text{ nonempty and bounded}$$

$$X(y) := \{x : g(x,y) \leq 0\} \text{ nonempty, closed and bounded for all } y \in S$$

so the convex function

$$\Phi(y) := \min_{x \in X(y)} f(x,y)$$

is well-defined for all $y \in S$.

Working on the y-space (projection)

$$\min_{y,x} f(x,y) = \min_{y} \min_{x} f(x,y)$$
$$g(x,y) \leq 0$$
$$Ay \leq b$$
$$y \text{ integer}$$

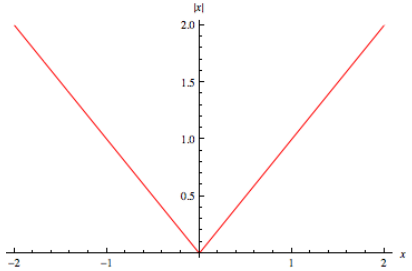
$$\Phi(y) := \min_{x} f(x,y')$$
$$g(x,y') \leq 0$$
$$(Ay' \leq b)$$
$$y' = y$$

$$\min_{y} \Phi(y)$$
$$Ay \leq b$$
$$y \text{ integer}$$

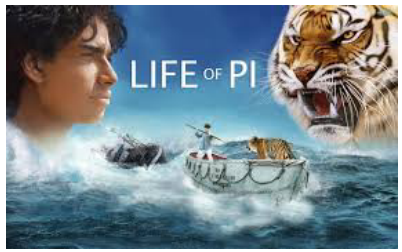
Original MINLP in the (x,y) space → **Master problem** in the y space

Warning: projection changes the objective function shape!

$$\min x$$
$$x \geq y$$
$$x \geq -y$$
$$y \in [-1,1]$$

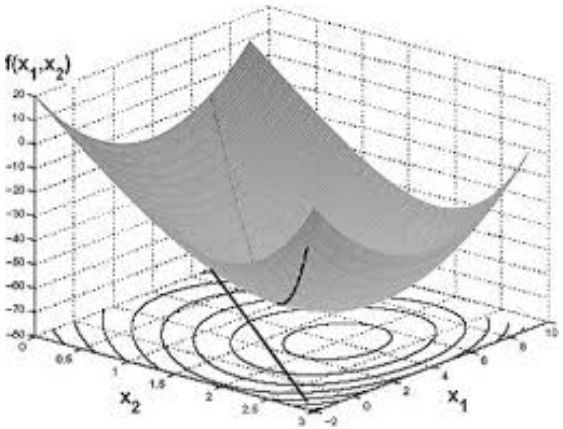


$$\min \Phi(y) = |y|$$
$$y \in [-1,1]$$

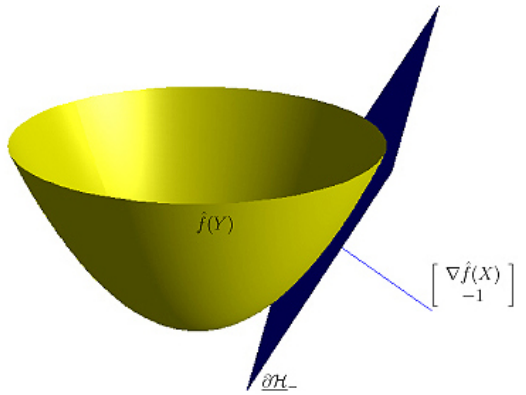


Life of P(H)I

- Solving Benders' master problem calls for the minimization of a **nonlinear** function **(even if you start from a linear problem!)**



- Branch-and-cut MINLP solvers generate a sequence of **linear cuts** to approximate this function from below (**outer-approximation**)



$$\begin{aligned} \min\ & w \\ \text{s.t. } & w \geq \Phi(y) \\ & Ay \leq b \\ & y \text{ integer} \end{aligned}$$

$$w \geq \Phi(y) \geq \Phi(y^*) + \xi(y^*)^T (y - y^*)$$

Benders cut computation

- **Benders** (for linear) and **Geoffrion** (general convex) told us how to compute a **(sub)gradient** to be used in the cut derivation, by using the optimal primal-dual solution *(x*,u*)* available after computing $\Phi(y^*)$

$$\xi(y^*) = \nabla_y f(x^*, y^*) + u^* \nabla_y g(x^*, y^*)$$

- This formula is **problem-specific** and perhaps **#scaring**
- By rewriting

$$\Phi(y^*) = \min\{f(x, \mathbf{q}) \mid g(x, \mathbf{q}) \leq 0,\ y^* \leq \mathbf{q} \leq y^*\}$$

we obtain a much **simpler recipe** to derive the same Benders cut:

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

#TheCurseOfKelley

• Master problem is typically solved by a **cutting plane method** where primal (fractional) solutions *y** and Benders cuts are generated on the fly

• A main reason for Benders' slow convergence is the use of **Kelley's** cutting plane recipe **"Always cut the optimal solution of the previous master"**

• In the first iterations, the master can contain too few constraints (sometimes, only variable bounds) → **zig-zagging** in the *y* space (lower bound stalling)

→**Stabilization** required as in

Column Generation and
Lagrangian Relaxation

e.g. through bundle methods

Escaping the #CurseOfKelley

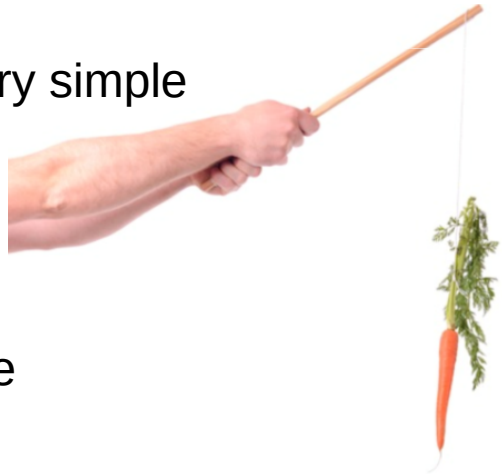
- Root node LP bound **very critical** → many ships sank here!

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

- Stabilization using "interior points"

(Ben-Ameur and Neto 2007, Fischetti and Salvagnin 2010, Naoum-Sawaya and Elhedhli 2013).

- For facility location problems, 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…

Our #ChaseTheCarrot heuristic

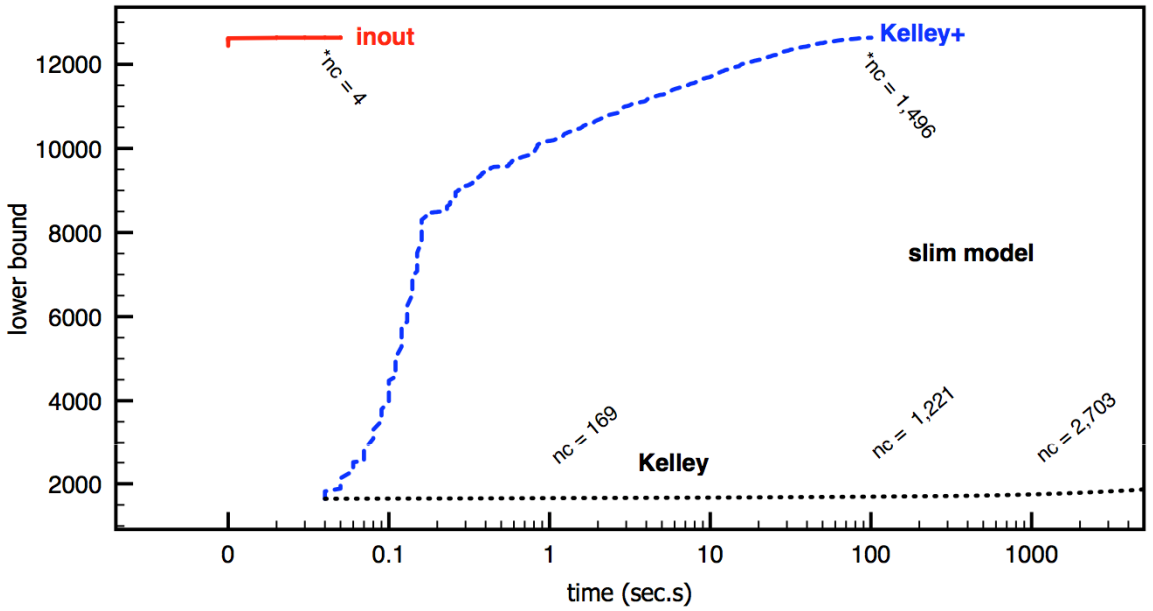
• We (the donkey) start with y = (1,1,…,1) and optimize the master LP as in Kelley, to get optimal y* (the carrot on the stick).

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

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

• **Warning: adaptations needed if feasibility Benders 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**

Thanks for your attention

- Full papers

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.

M. Fischetti, I. Ljubic, M. Sinnl, "Benders decomposition without separability: a computational study for capacitated facility location problems", Tech. Rep. UniPD, 2015.

and slides available at

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