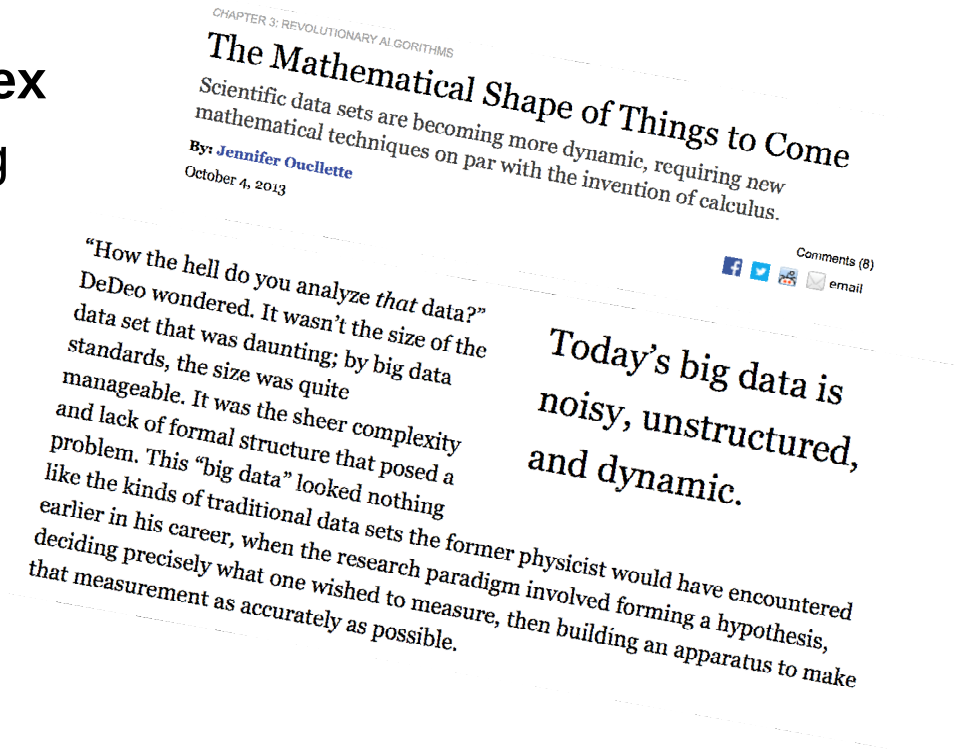
Thin Models for Big Data

Matteo Fischetti, University of Padova

Big Data

- **Big data** is a broad term for data sets so **large** and **complex** that traditional data processing applications are inadequate



CHAPTER 3: REVOLUTIONARY ALGORITHMS

The Mathematical Shape of Things to Come

Scientific data sets are becoming more dynamic, requiring new mathematical techniques on par with the invention of calculus.

By: Jennifer Ouellette

October 4, 2013

Comments (8) email

"How the hell do you analyze *that* data?" DeDeo wondered. It wasn't the size of the data set that was daunting; by big data standards, the size was quite manageable. It was the sheer complexity and lack of formal structure that posed a problem. This "big data" looked nothing like the kinds of traditional data sets the former physicist would have encountered earlier in his career, when the research paradigm involved forming a hypothesis, deciding precisely what one wished to measure, then building an apparatus to make that measurement as accurately as possible.

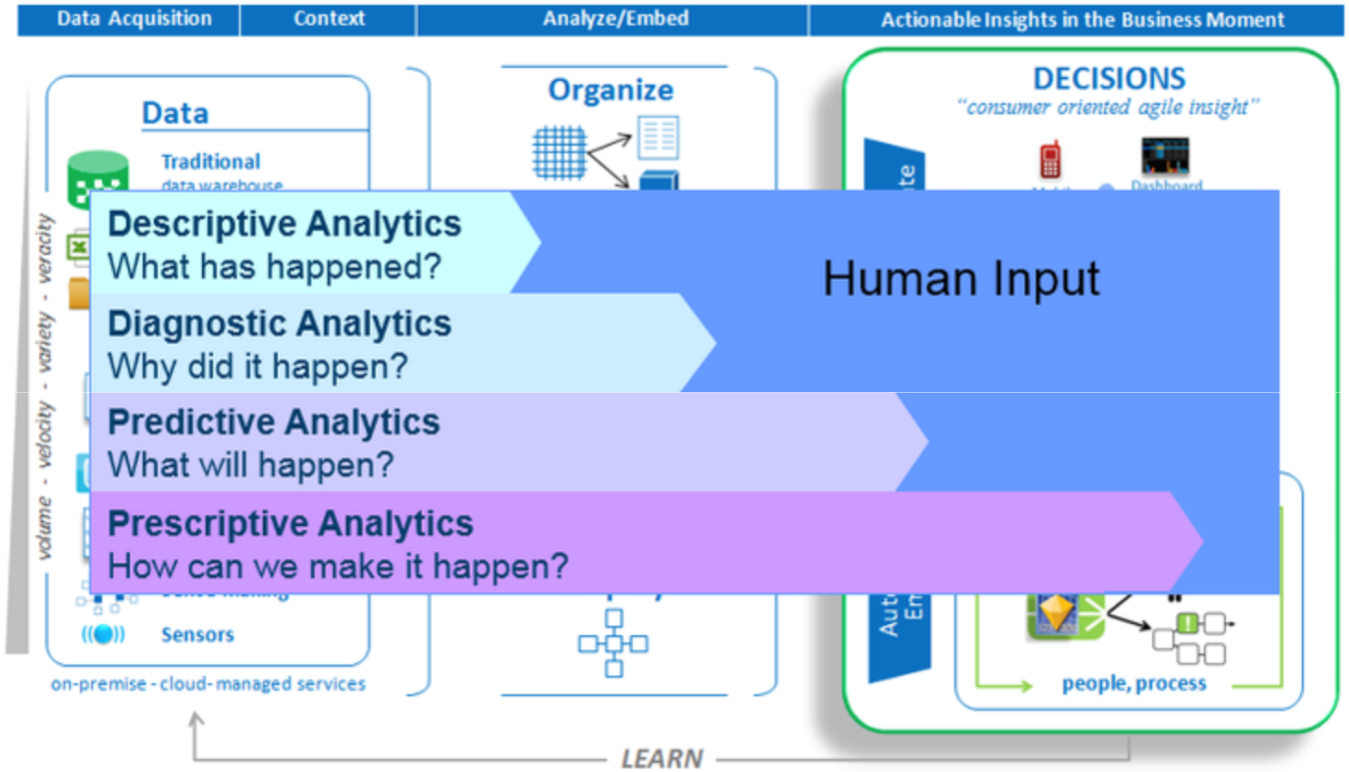
Today's big data is noisy, unstructured, and dynamic.

- **Multimodal** (incomplete) data from **any kind of sources** (physical/social systems etc.)

- Not just a matter of **size** …

- We (**OR/Mathematical Optimization** people) are already proud if we can solve problems with few MB's of **perfectly clean** input data... **#TooBigForUs?**

Big Data in the realm of Analytics

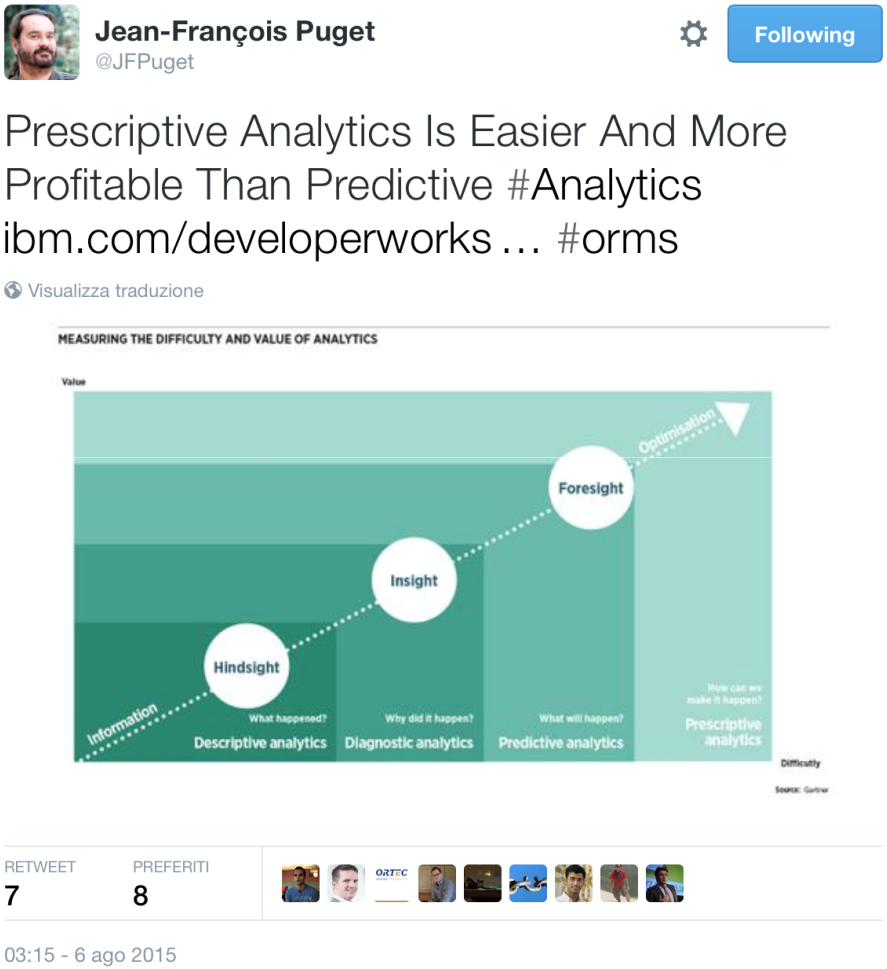


Jean-François Puget @JFPuget · 16 ago

The Future of #Analytics Is Prescriptive, Not Predictive clickz.com/2416723 #orms Well said.

17 17

#WeArePrescriptive



It turns out that

Prescriptive Analytics

~=

(large scale)

Mathematical Optimization

we are

no longer old-fashioned

but actually have a LOT to say about Big Data

But … are we “scalable enough”?

• Prescriptive Analytics calls for **scalable** methods to deal with **larger and larger problems**

• Operations Research / Mathematical Optimization models and algorithms can be inherently rather **sophisticated**

• **Challenge: improve scalability of OR heuristic and exact methods**

• **Simplicity** is one of keys to scalability → focus (as much as possible) on simple models and solution schemes

• Warning: **simple** does not mean trivial ...

Occam’s razor

- **Occam's razor**, or law of parsimony (lex parsimoniae): a problem-solving principle devised by the English philosopher William of Ockham (1287–1347).
- Among competing hypotheses, the one with the fewest assumptions is more likely be true and should be preferred—the fewer assumptions that are made, the better.
- **The simpler (the model, the algorithm) the better**
- Used as a heuristic guide in the development of theoretical models in physics (Albert Einstein, Max Planck, Werner Heisenberg, etc.)
- Not to misinterpreted and used as an excuse to address oversimplified models: “**Everything should be kept as simple as possible, but no simpler**” (Albert Einstein)

Thinning out optimization models

- The practical difficulty in solving hard problems sometimes comes for **overmodelling**:

  Too many vars.s and constr.s just

  suffocate the model

  (and the cure is not to complicate it even more!)

  Let your model breathe!

- **Simpler** and more effective models can sometimes be obtained by:
  1. Choosing a model that better fits the **instances** of interest
  2. Removing variables that play a little role for the **problem class** of interest
  3. Using **decomposition** to break the problem into smaller pieces

Example 1: QAP

- **Quadratic Assignment Problem** (QAP): extremely hard to solve

- Unsolved **esc*** instances from QAPLIB (attempted on constellations of thousand computers around the world for many CPU years, with no success)

- The thin out approach: *esc* instances are
  - very large → use **slim MILP** models with high node throughput
  - decomposable → solve pieces **separately**
  - very symmetrical → find a cure and **simplify** the model through ***Orbital Shrinking*** to actually **reduce** the size of the instances

Fischetti, L. Liberti, "Orbital shrinking", Lecture Notes in Computer Science, Vol. 7422, 48-58, 2012.

- **Outcome**:

  a. all esc* but two instances solved in **minutes** on a notebook

  b. **esc128** (by far the largest QAP ever attempted) solved in just **seconds**

M. Fischetti, M. Monaci, D. Salvagnin, "Three ideas for the Quadratic Assignment Problem", Operations Research 60 (4), 954-964, 2012.

Example 2: Steiner Trees

- Recent **DIMACS 11 (2014)** challenge on Steiner Tree Problems: various versions and categories (exact/heuristic/parallel/…) and scores (avg/formula 1/ …)

- Standard MILP models use *x* var.s (arcs) and *y* var.s (nodes)

- Many very hard (unsolved) instances available on STEINLIB
- **Observation**: many hard instances have uniform arc costs

- **Thin out**: remove *x* var.s and work on the *y*-space (kind of Benders' projection)
- Heuristics based on the **blur** principle: initially forget about details…

- Vienna-Padua team → **MozartBalls** code
- **Outcome:**
  - Some open instances solved in a few **seconds**
  - MozartBalls ranked first in most DIMACS categories

M. Fischetti, M. Leitner, I. Ljubic, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, M. Sinnl, "Thinning out Steiner trees: a node-based model for uniform edge costs", Tech.Rep., 2014

Example 3: facility location

- **Uncapacitated facility location** with linear (UFL) and quadratic (qUFL) costs
- **Huge** MILP models involving y var.s (selection) and x var.s (assignment)
- **Thin out**: assignment var.s *x* **suffocate** the model, just remove them…
- A perfect fit with **Benders decomposition** (more later!)
- **Outcome**:
  - Many hard UFL instances solved in just **seconds** on a notebook
  - Seven open instances solved to optimality, 22 best-known improved
  - Speedup of **4 orders of magnitude** for qUFL up to size 150x150
  - Solved qUFL instances up to 2,000 locations and 10,000 clients in just 5 minutes **(MIQCP's with 20M SOC constraints and 40M var.s)**

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

Thin out your favorite model

call Benders toll free

Benders decomposition well known

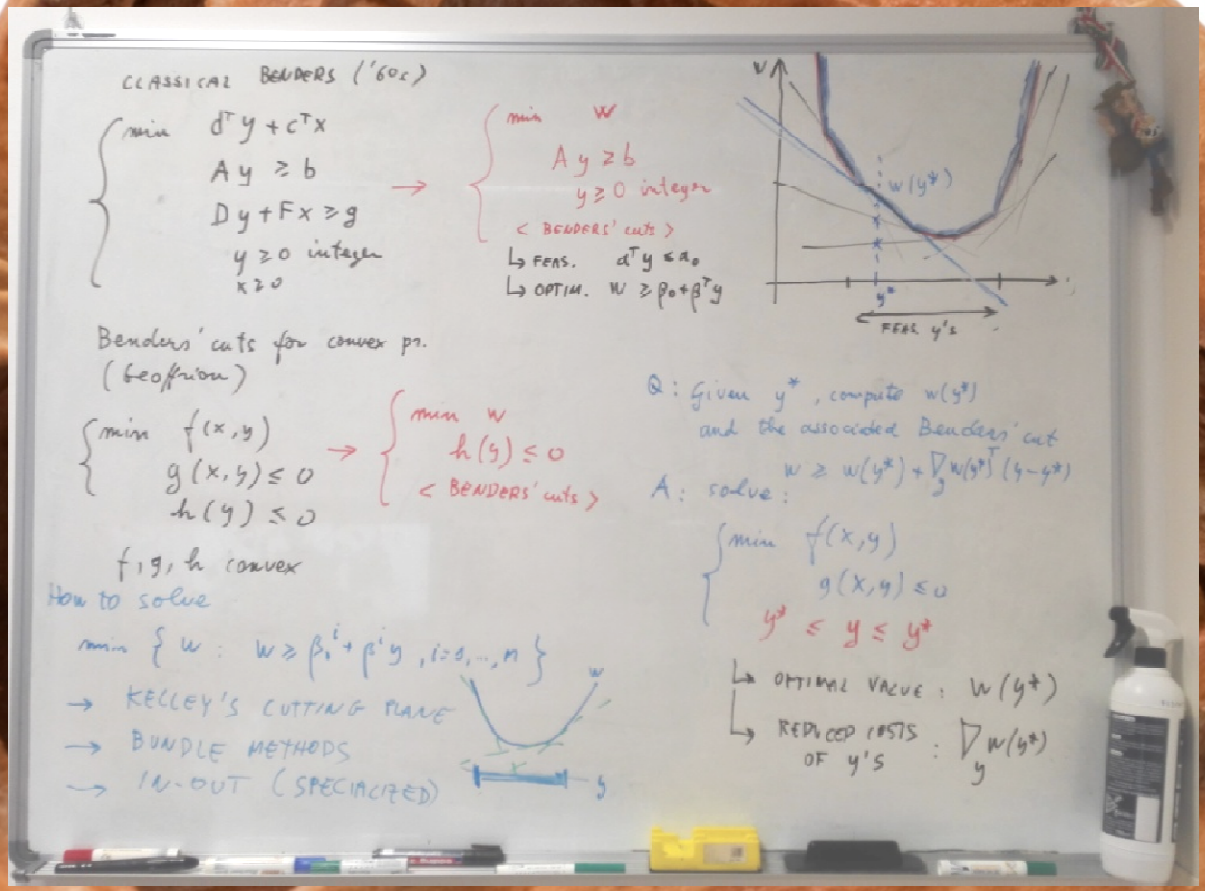
… but not so many MIPeople actually use it

We will next give a brief tutorial on Modern Benders

hoping to convince
young researchers
to test it…

…and not to be #TooBoring

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)

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

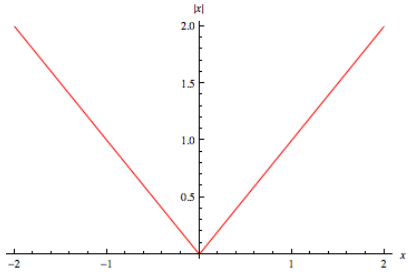
$$\Phi(y) := \min_{x \in X(y)} f(x,y)$$

$$\begin{aligned} \min\ & \Phi(y) \\ Ay &\le b \\ y\ & \text{integer} \end{aligned}$$

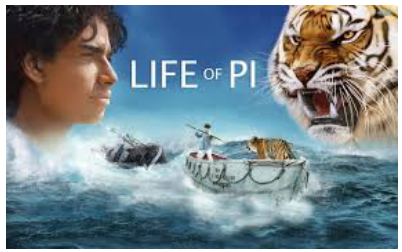
Original MINLP in the (x,y) space → **Master proble**m in the y space

Warning: projection changes the objective function shape!

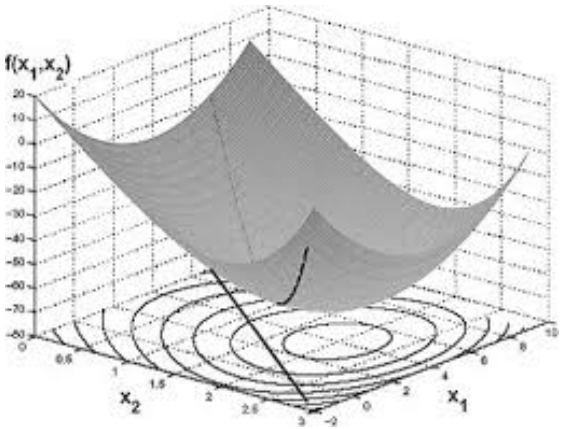
$$\begin{aligned} \min\ & x \\ x &\ge y \\ x &\ge -y \\ y &\in [-1,1] \end{aligned}$$



$$\begin{aligned} \min\ & \Phi(y) = |y| \\ y &\in [-1,1] \end{aligned}$$

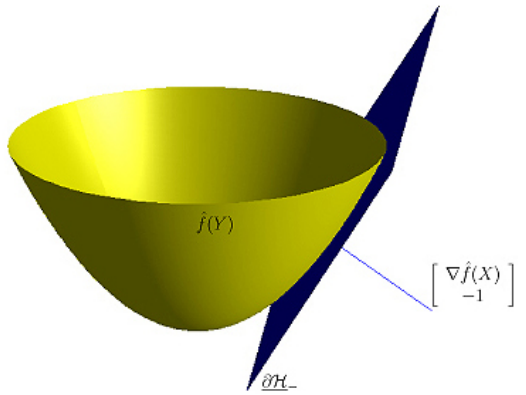


Life of P(H)I



- 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

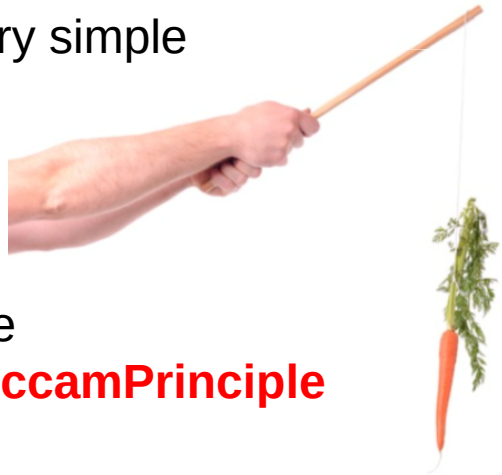
- Root node LP bound **very critical** → many ships sank here!

- **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 **#OccamPrinciple**

Our #ChaseTheCarrot heuristic

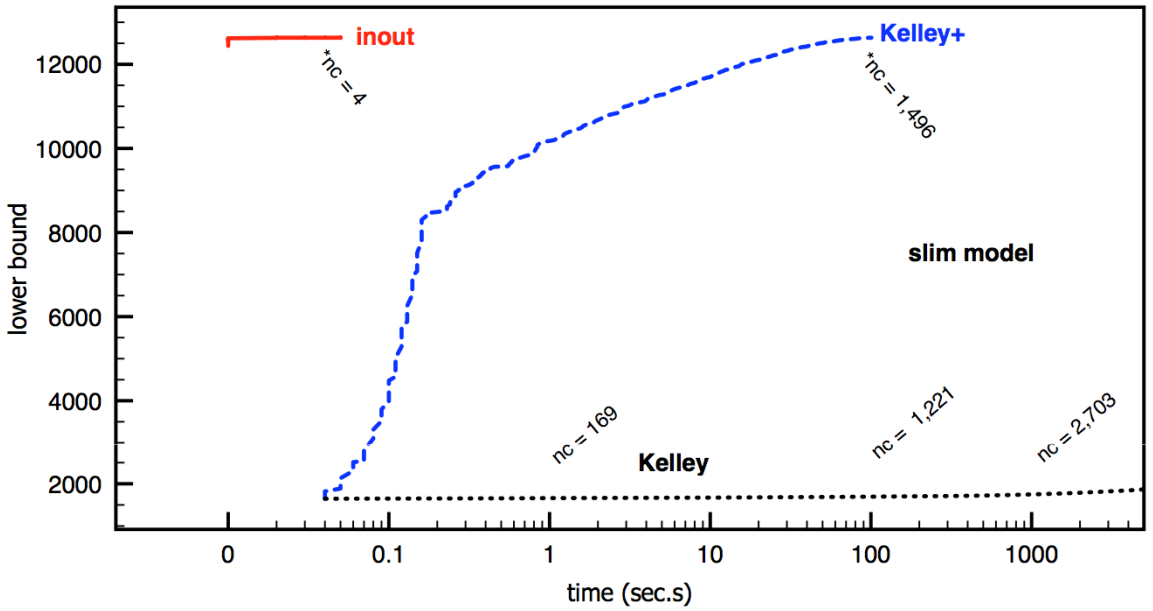
• We (the donkey) start with y = (1,1,…,1) and optimize the master LP as in Kelley, to get optimal y* (the carrot on the stick).

• 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**

Conclusions

- **Operations Research (OPS! Prescriptive Analytics) can play an important role in the Datazoic era …**

  … giving us an incentive to design simpler and more scalable solvers

  … possibly based on clever decomposition

- **Benders is a good boy, if you do not mistreat him!**

Thanks for your attention!

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

(not too) Big Data and OR as good friends