Branch-and-Cut is our swiss army knife

Matteo Fischetti, University of Padova

Branch & Cut ™

SIAM REVIEW
Vol. 33, No. 1, pp. 60–100, March 1991

004

A BRANCH-AND-CUT ALGORITHM
FOR THE RESOLUTION OF LARGE-SCALE
SYMMETRIC TRAVELING SALESMAN PROBLEMS *

MANFRED PADBERG† AND GIOVANNI RINALDI‡

- A “trademark” by Manfred Padberg and Giovanni Rinaldi
- Proposed in the 1990’s for the TSP (and soon extended)
- Comes as an **algorithm** entangled with its **implementation**

Conjecture: *Using cuts within an enumerative scheme is good*.

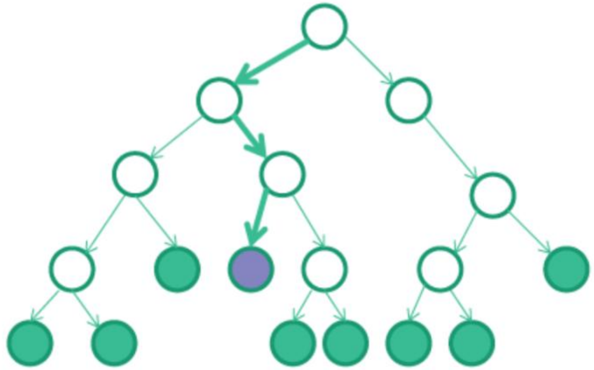
Proof. Assume w.l.o.g. a good LP solver. Then apply B&Bound but

- make use of families of (problem dependent) globally-valid inequalities
- perform efficient exact/heuristic cut separation on the fly
- use a data-structure (cut pool) to effectively share cuts among nodes
- price variables in a dynamic way (well before branch-and-price!)
- alternate row and column generation in a sound way …
- suspend a node if “unattractive”
- …

Modern B&C implementation

- Modern B&C solvers such as Cplex, Gurobi, Express, SCIP etc. can be fully **customized** by using ***callback functions***

- Callback functions are just **entry points** in the B&C code where an advanced user (you!) can add his/her customizations



- Most-used callbacks (using old-style Cplex's jargon)
  - **Lazy constraint**: add "lazy constr.s" that should be part of the original model
  - **User cut**: add additional contr.s that hopefully help enforcing feasibility/integrality
  - Heuristic: try to improve the incumbent (primal solution) as soon as possible
  - Branch: modify the branching strategy
  - …

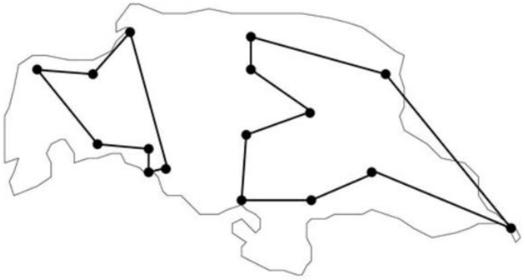
Lazy constraint callback
CPX_CALLBACKCONTEXT_CANDIDATE

- Automatically invoked when a solution is going to update the **incumbent** (meaning it is **integer** and **feasible** w.r.t. current model, often coming from an internal primal heuristic)

- This is the **last checkpoint** where we can discard a solution for whatever reason (e.g., because it violates a constraint that is not part of the current model)

- To avoid be bothered by this solution again and again, we can/should return a **violated constraint (cut)** that is added (globally or locally) to the current model

- Cut generation is often **simplified** by the fact that the solution to be cut is known to be **integer** (e.g., SECs for TSP)



Usercut callback

CPX_CALLBACKCONTEXT_RELAXATION

- Automatically invoked at every B&B node when the current solution is **noninteger** (e.g., just before branching)

- A **violated cut** can possibly be returned, to be added (locally or globally) to the current model → often leads to an improved convergence to integer solutions

- If no cut is returned, **branching** occurs as usual

- Cut generation **can be hard** as the point is noninteger (heuristic approaches can be used)

- User cuts are **not mandatory** for B&C correctness → being too clever on them can actually **slow-down** the solver because of the overhead in generating and using them (larger/denser LPs etc.)

Other callbacks

- **Branch callback**: invoked at the end of each node (even when the LP solution is integer and apparently does not require any cut/branching) and used to impose/customize branching

- **Heuristic callback**: used to build new (possibly problem-specific) feasible integer solutions to be **posted**, i.e., passed to the solver which will use them (at the appropriate time) to possibly update the incumbent

- etc. etc.

But… how do we generate cuts?

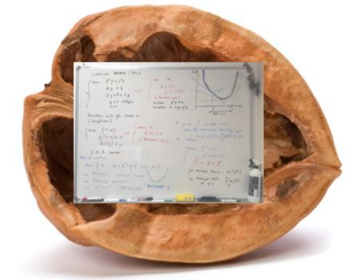
- **Problem-specific cuts**

- **General-purpose** MIP cuts trying to enforce integrality (e.g. Gomory cuts)
  - Modern solvers already have a lot of them, implemented in the most effective way…

- Cuts coming from an alternative **extended formulation**
  - **Benders cuts**
  - **…**

- Cuts dealing with nonlinearities
  - **Intersection cuts** (e.g., for **bilevel optimization)**
  - …

Benders’ cuts for dummies



- The original Benders’ paper from the ‘60s uses **two** distinct ingredients for solving a Mixed-Integer Linear Program (MILP):

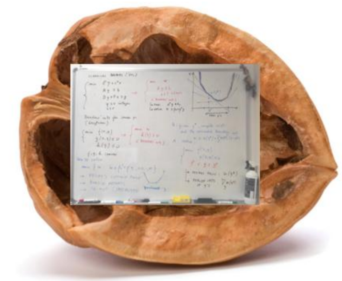
  1) A **cut loop strategy** where a relaxed **(NP-hard)** MILP is solved exactly (i.e., to **integrality**) by a black-box solver, and then is iteratively tightened by means of additional **“Benders” linear cuts**

  2) The **technicality** of how to actually compute those cuts (Farkas’ projection)

  – Papers proposing “a new Benders-like scheme” typically refer to 1)

  – Students scared by “Benders implementations” typically refer to 2)

Benders cuts for dummies



- **Later developments** in the '70s
  - Folklore (Miliotios for TSP?)
    - generate Benders cuts within a **single B&B tree** to cut any infeasible integer solution that is going to update the incumbent

    → **lazycuts**
  - McDaniel & Devine (1977)
    - use Benders cuts to cut **fractional sol.s** as well

    → **usercuts**
- Everything fits very naturally within a modern **Branch-and-Cut** (B&C) framework!

Modern Benders

- Consider the **convex** MINLP in the *(x,y)* space

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

Warning: the important var.s are the y's !

- For the sake of simplicity, assume that:
  - the set $S := \{y : Ay \le b\}$ is nonempty and bounded
  - the **convex function**

$$\Phi(y) := \min_x f(x,y)$$
$$g(x,y) \le 0$$

is well defined for all y ∈ S

Working on the y-space (projection)

(1)

$$\min_y \min_x f(x,y)$$
$$g(x,y) \le 0$$
$$Ay \le b$$
$$y \text{ integer}$$

(2)

"isolate the inner minimization over *x*"

$$\Phi(y) := \min_x f(x,y)$$
$$g(x,y) \le 0$$

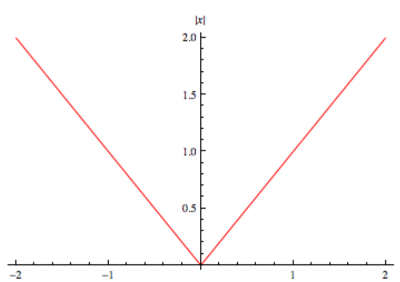
(3)

$$\min \Phi(y)$$
$$Ay \le b$$
$$y \text{ integer}$$

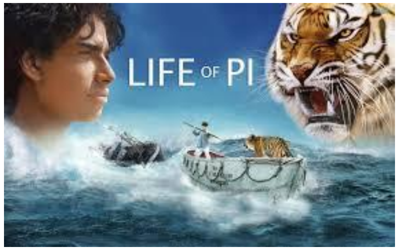
Original MINLP in the (x,y) space → Benders' **master** problem in the *y* space

Warning: projection changes the objective function (e.g., linear → convex nonlinear)

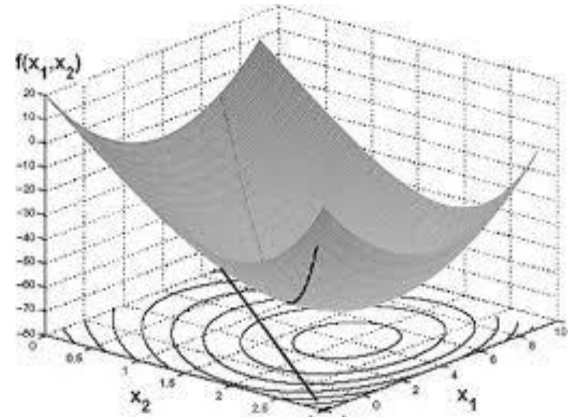
$$\min x$$
$$x \ge y$$
$$x \ge -y$$
$$y \in [-1,1]$$



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



Life of P(H)I

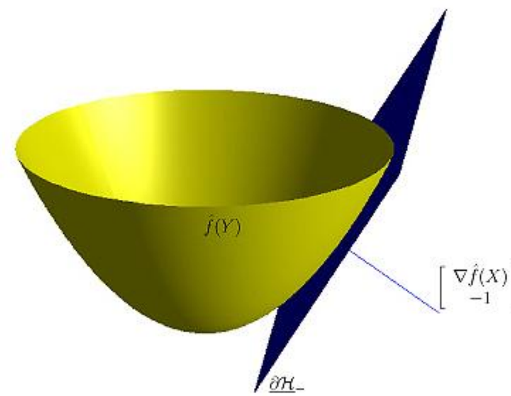


- Solving Benders' master problem calls for the minimization of a **nonlinear** convex function **(even if you start from a linear problem!)**

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

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

subgradient
(aka Benders) cut →



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

Benders cut computation

- **Benders** (for linear) and **Geoffrion** (general convex) use Linear/Lagrangian duality to compute a **subgradient** to be used in the cut derivation

- Given an optimal primal-dual solution *(x*,u*)* available after computing $\Phi(y^*)$, a subgradient of $\Phi(y)$ in *y** is computed as

$$\Phi(y) := \min_x f(x,y) \quad g(x,y) \le 0 \quad \rightarrow \quad \xi(y^*) = \nabla_y f(x^*,y^*) + u^* \nabla_y g(x^*,y^*)$$

- The above formula is **problem-specific** and sometimes cumbersome

- This is maybe the reason why Benders cuts are considered “too sophisticated” by students

1-2-3: Benders!

- But … you can kindly ask your solver to make all calculations for you!

$$\min f(x,y)$$
$$g(x,y) \leq 0$$
$$Ay \leq b$$

- Here is the **recipe**:

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 feasibility cuts

- For some important applications, the set

$$X(y) := \{x : g(x,y) \le 0\}$$

can be empty for some "**infeasible**" $y \in S$

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

- This situation can be handled by considering the "phase-1" feasibility condition

$$0 \ge \Psi(y) := \min\{1^T s \,|\, g(x,y) \le s,\ s \ge 0\}$$

where the function $\Psi(y)$ is **convex**
→ it can be approximated by the usual subgradient **"Benders feasibility cut"**

$$0 \ge \Psi(y) \ge \Psi(y^*) + \xi(y^*)^T (y - y^*)$$

to be computed using reduced costs as before

Successful Benders applications

- Benders cuts work well when fixing *y* = *y** for computing $\Phi(y^*)$ makes the problem **much simpler to solve**.

- This usually happens when
  - The problem for *y* = *y** decomposes into a number of **independent subproblems**
    - Stochastic Programming
    - Uncapacitated Facility Location
    - etc.

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

  - Fixing *y* = *y** **changes the nature** of some constraints:
    - in Capacitated Facility Location, tons of constr.s of the form $x_{ij} \le y_j$ become just variable bounds
    - Second Order Constraints $x_{ij}^2 \le z_{ij}\, y_i$ become quadratic constr.s
    - etc.

Benders cuts instability

- B&C codes generate cuts, on the fly, in a **sequential** fashion
- Consider e.g. the **root B&C node** (arguably, the most critical one)
- A classical **cut-loop scheme** (described here for MILPs)

  J. E. Kelley. The cutting plane method for solving convex programs, Journal of the SIAM, 8:703-712, 1960.

  – Find an optimal **vertex** y^* of the current LP relaxation

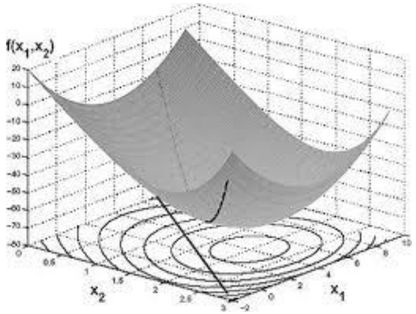
  – Invoke a separation function on y^*, add the returned violated cut (if any) to the current LP, and repeat

Benders cuts instability

- This cut loop can be very **ineffective** in the **first iterations** when few Benders cuts have been generated, and the system A y ≤ b contains just few constraints (often, only variable bounds)

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

- In this situation:
  - the current master sol. *y** is almost unconstrained
    → **zig-zagging phenomenon**
  - Benders cuts convey information around “erratic” points *y** far from the region of interest



→ **Stabilization is** needed, e,g. through **Frank-Wolfe** cut loop

Conclusions

To summarize:

- Benders cuts are **easy** to implement within modern B&C (just use a callback where you solve the problem for $y = y^*$ and compute reduced costs)

- Kelley's cut loop can be **desperately slow** hence stabilization is a **must**

Benders implemented in CPLEX **general** MIP solver since version 12.7

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

Reference papers:

M. Fischetti, I. Ljubic, M. Sinnl, "Benders decomposition without separability: a computational study for capacitated facility location problems", European Journal of Operational Research, 253, 557-569, 2016.

M. Fischetti, I. Ljubic, M. Sinnl, "Redesigning Benders Decomposition for Large Scale Facility Location", Management Science 63 (7), 2146-2162, 2017.