Intersection Cuts from Bilinear Disjunctions

Matteo Fischetti, University of Padova
(joint work with Michele Monaci, University of Bologna)

Non-convex MIQP

- **Goal: implement a Mixed-Integer (non-convex) Quadratic solver**

- Two approaches:

  1. start with a continuous QP solver and add enumeration on top of it
  → implement B&B to handle integer var.s

  2. start with a MILP solvers (B&C) and customize it to handle the
  non-convex quadratic terms → add McCormick & spatial branching

     PROS: …

     CONS: …

- This talk goes for 2.

MIQP as a MILP with bilinear eq.s

- The fully-general MIQP of interest reads

$$
\begin{array}{lll}
(MIQP) & \min a_0^T x + x^T Q^0 x & \\
& a_k^T x + x^T Q^k x \,@\, b, & k = 1, \ldots, m \\
& \ell_j \le x_j \le u_j, & j = 1, \ldots, n \\
& x_j \text{ integer}, & j \in \mathcal{I}, \\
& x_j \text{ continuous}, & j \in \mathcal{C},
\end{array}
$$

and can be restated as

$$
\begin{array}{lll}
(MIBLP) & \min_x c^T x & \\
& Ax = b & \\
& \ell_j \le x_j \le u_j, & j = 1, \ldots, n \\
& x_j \text{ integer}, & j \in \mathcal{I} \\
& x_j \text{ continuous}, & j \in \mathcal{C} \\
& x_{r_k} = x_{p_k} x_{q_k}, & k = 1, \ldots, K,
\end{array}
$$

McCormick inequalities

- To simplify notation, rewrite the generic bilevel eq. $x_{r_k} = x_{p_k} x_{q_k}$ as:

$$z = x\,y$$
$$\ell_x \le x \le u_x$$
$$\ell_y \le y \le u_y$$

- Obviously

$$\begin{array}{rcll}
(x-\ell_x)(y-\ell_y) \ge 0 & & \mathrm{mc1)} & z \ge \ell_y x + \ell_x y - \ell_x \ell_y \\
(x-u_x)(y-u_y) \ge 0 & \rightarrow & \mathrm{mc2)} & z \ge u_y x + u_x y - u_x u_y \\
(x-\ell_x)(y-u_y) \le 0 & & \mathrm{mc3)} & z \le u_y x + \ell_x y - \ell_x u_y \\
(x-u_x)(y-\ell_y) \le 0 & & \mathrm{mc4)} & z \le \ell_y x + u_x y - u_x \ell_y
\end{array}$$

(just replace *xy* by *z* in the products on the left)

- Note: *mc1)* and *mc2)* can be improved in case *x=y* → **gradients cuts**

$$z \ge x_0^2 + 2x_0(x - x_0), \quad \text{for each } x_0 \in \Re$$

Spatial branching

- McCormick inequalities are not perfect
  → they are tight only when *x* and/or *y* are at their lower/upper bound

$$(x-\ell_x)(y-\ell_y) \geq 0$$
$$(x-u_x)(y-u_y) \geq 0$$
$$(x-\ell_x)(y-u_y) \leq 0$$
$$(x-u_x)(y-\ell_y) \leq 0$$

→ at some B&C nodes, it may happen that the current (fractional or integer) solution satisfies **all** MC inequalities but some bilinear eq.s $z = xy$ are still violated (we call this **#bilinear_infeasibility**)

→ we need a **bilinear-specific branching** (the usual MILP branching on integrality does not work if all var.s are integer already)

- **Standard Spatial Branching**: if $z^* = x^* y^*$ is violated, branch on

(x ≤ x*) OR (x ≥ x*)

to make the upper (resp. lower) bound on *x* tight at the left (resp. right) child node – thus improving the corresponding MC inequality

A new branching rule

- **Shifted Spatial Branching**: let $\rho^* := z^* - x^* y^*$; *if* $\rho^* > 0$, branch on

$$(x \le x^* - \delta) \text{ OR } (x \ge x^* - \delta)$$

where $\boldsymbol{\delta}$ is defined so as to **balance** the violation of the two child nodes (case $\rho^* < 0$ is similar)

- **Left branch** $(u_x = x^* - \delta) \rightarrow$ *violation of* $\boldsymbol{\delta}$ *of the upper bound* u_x

- **Right branch** $(l_x = x^* - \delta) \rightarrow$ *violation of* $\boldsymbol{\delta}$ *for the MC ineq.* $(x - x^* + \delta)(y - u_y) \le 0$ by choosing $\delta = \rho^*/(1 + u_y - y^*)$

- **New Branching Rule**: among all violated $z^* = x^* y^*$, select the one maximizing the balanced violation $\boldsymbol{\delta}$

The branching procedure

Algorithm 1: Our branching procedure

Input : The bilinear-infeasible point x^* and the variable bounds $(\overline{\ell}, \overline{u})$ at the current node; the tolerance value ε for constraint violation;

Output: The branching variable $x_{\texttt{bvar}}$ and the corresponding threshold value θ for spatial branching;

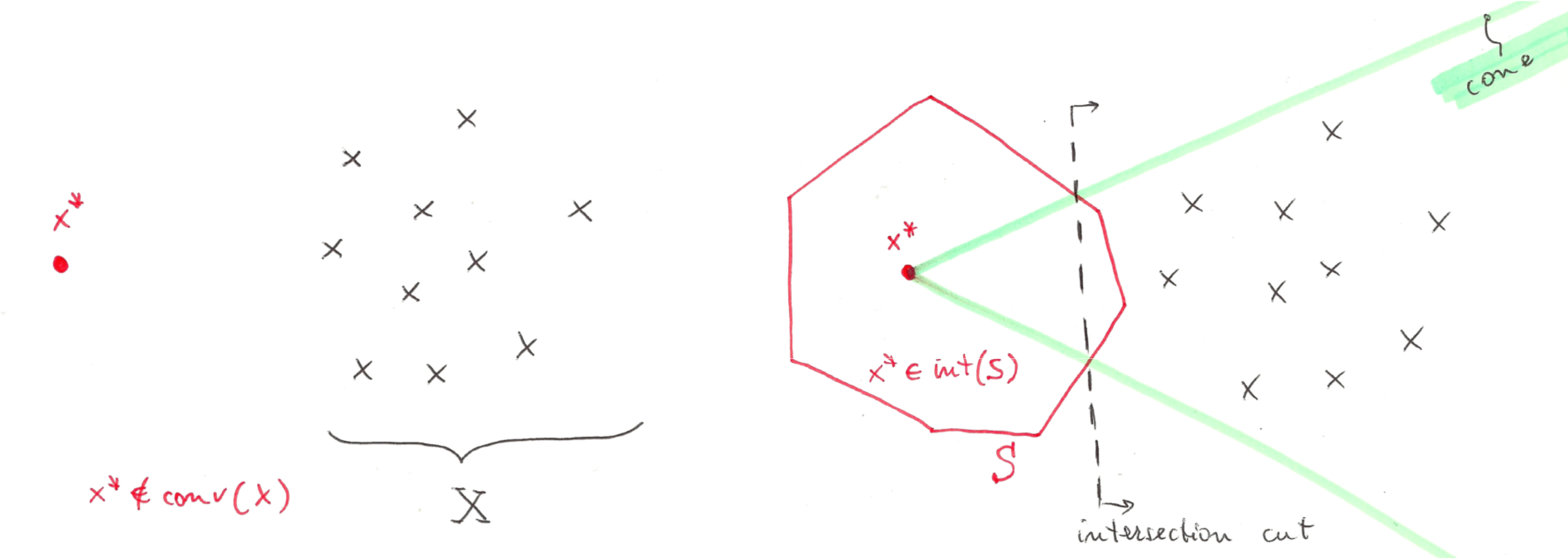
$\delta := -\infty$; $\texttt{bvar} := -1$; $\theta := 0$;
for *each* $k \in \{1, \ldots, K\}$ *with* $|x^*_{r_k} - x^*_{p_k} x^*_{q_k}| > \varepsilon$ **do**
    $\texttt{branch\_score}(x^*,\ \overline{\ell},\ \overline{u},\ p_k,\ q_k,\ z_k,\ \texttt{bvar},\ \delta,\ \theta)$;
    $\texttt{branch\_score}(x^*,\ \overline{\ell},\ \overline{u},\ q_k,\ p_k,\ z_k,\ \texttt{bvar},\ \delta,\ \theta)$;
end
if $(\theta < \overline{\ell}_{\texttt{bvar}} + \varepsilon$ **or** $\theta > \overline{u}_{\texttt{bvar}} - \varepsilon)$ **then** $\theta := (\overline{\ell}_{\texttt{bvar}} + \overline{u}_{\texttt{bvar}})/2$ **endif**;
return $(\texttt{bvar}, \theta)$;

Algorithm 2: **function** $\texttt{branch\_score}(x^*,\ \overline{\ell},\ \overline{u},\ ix,\ iy,\ iz,\ \texttt{bvar},\ \delta,\ \theta)$

$\rho^* = x^*_{iz} - x^*_{ix} x^*_{iy}$;
if $(\rho^* < -\varepsilon/2)$ **then** // x^*_{iz} is too small: use McCormick ineq.s mc1 or mc2 to increase it
    $d := -\rho^*/(1 + x^*_{iy} - \overline{\ell}_{iy})$;
    if $(d > \delta)$ **then** $\texttt{bvar} := ix$, $\theta := x^*_{ix} - d$, $\delta := d$ **endif**;
    $d := -\rho^*/(1 + \overline{u}_{iy} - x^*_{iy})$;
    if $(d > \delta)$ **then** $\texttt{bvar} := ix$, $\theta := x^*_{ix} + d$, $\delta := d$ **endif**;
else // x^*_{iz} is too large: use McCormick ineq.s mc3 or mc4 to reduce it
    $d := \rho^*/(1 + \overline{u}_{iy} - x^*_{iy})$;
    if $(d > \delta)$ **then** $\texttt{bvar} := ix$, $\theta := x^*_{ix} - d$, $\delta = d$ **endif**;
    $d := \rho^*/(1 + x^*_{iy} - \overline{\ell}_{iy})$;
    if $(d > \delta)$ **then** $\texttt{bvar} := ix$, $\theta := x^*_{ix} + d$, $\delta := d$ **endif**;
end

Intersection Cuts (ICs)

- **Intersection cuts** (Balas, 1971): a powerful tool to separate a point **x*** from a set **X** by a liner cut



- All you need is (love, but also)
  - a **cone** pointed at **x*** containing all **x ε X**
  - a **convex set S** with x* (but no **x ε X**) in its <u>**interior**</u>
- If x* **vertex** of an LP relaxation, a suitable cone comes for the **LP basis**

Bilinear-free sets

- **Observation**: given an infeasible point x*, any branching disjunction violated by x* implicitly defines a **convex set S** with x* (but no feasible x) in its **interior**

$$\bigvee_{i=1}^{k} (g_i^T x \geq g_{i0}) \quad \rightarrow \quad S = \{x : g_i^T x \leq g_{0i}, \ i = 1, \ldots, k\}$$

- Thus, **in principle,** one could always generate an IC instead of branching → not always advisable because of numerical issues, slow convergence, tailing off, cut saturation, etc. **#LikeGomoryCuts**

- **Candidate branching disjunctions** (supplemented by MC cuts) are the 1- and 2-level (possibly shifted) spatial branching conditions:

$$(x \leq x^*) \vee (x \geq x^*)$$

$$(x \leq x^*, y \leq y^*) \vee (x \leq x^*, y \geq y^*) \vee (x \geq x^*, y \leq y^*) \vee (x \geq x^*, y \geq y^*)$$

IC separation issues

- IC separation can be probematic, as we need to read the cone rays from the LP tableau → **numerical accuracy** can be a big issue here!

- For **MILP**s, ICs like Gomory cuts are **not mandatory** (so we can skip their generation in case of numerical problems), but for **MIBLP**s they are more instrumental **#SeparateOrPerish**

- **Notation**: consider w.l.o.g. an LP in standard form and no var. ub's

$\min\{\hat{c}^T \xi : \hat{A}\xi = \hat{b}, \xi \geq 0\}$ be the LP relaxation at a given node

$S = \{\xi : g_i^T \xi \leq g_{0i},\ i = 1, \ldots, k\}$ be the bilevel-free set

$\bigvee_{i=1}^{k} (g_i^T \xi \geq g_{i0})$ be the disjunction to be satisfied by all feas. sol.s

Numerically safe ICs

A **single** valid inequality can be obtained by taking, for each variable, the worst LHS Coefficient (and RHS) in each disjunction

To be applied to a **reduced form** of each disjunction where the coefficient of all basic variables is zero (kind of LP reduced costs)

$$\bigvee_{i=1}^{k} (g_i^T \xi \geq g_{i0})$$

$$\bigvee_{i=1}^{k} (\overline{g}_i^T \xi \geq \overline{g}_{i0})$$

$$\bigvee_{i=1}^{k} (\frac{\overline{g}_i^T}{\overline{g}_{i0}} \xi \geq 1)$$

Algorithm 1: Intersection cut separation

Input : An LP vertex ξ^* along with its a associated LP basis $\hat{B}$;
the feasible-free polyhedron $S = \{\xi : g_i^T \xi \leq g_{0i},\ i = 1, \ldots, k\}$ and the associated valid disjunction $\bigvee_{i=1}^{k} (g_i^T \xi \geq g_{i0})$ whose members are violated by ξ^*;

Output: A valid intersection cut violated by ξ^*;

for $i := 1$ **to** k **do**
  $(\overline{g}_i^T, \overline{g}_{i0}) := (g_i^T, g_{i0}) - u_i^T (\hat{A}, \hat{b})$, where $u_i^T = (g_i)_{\hat{B}}^T \hat{B}^{-1}$
end
for $j := 1$ **to** n **do** $\gamma_j := \max\{\overline{g}_{ij} / \overline{g}_{i0} : i \in \{1, \ldots, k\}\}$;
return *the violated cut* $\gamma^T \xi \geq 1$

B&C implementation

- Implementation using **IBM ILOG Cplex 12.8** using callbacks:
  - **Lazy constraint callback**: separation of MC inequalities for integer sol.s
  - **Usercut callback**: not needed (and sometimes detrimental)
  - **Branch callback:** our new spatial branching
  - **Incumbent callback**: very-last resort to kill a bilinear-infeasible integer solution (when everything else fails e.g. because of tolerances)
  - **MILP heuristics** (kindly provided by the MILP solver): active at their default level
  - **MIQP-specific heuristics**: not implemented yet

Computational analysis

- **Three** algorithms under comparison

  - ✓ **SCIP**: the general-purpose solver SCIP (vers. 5.0.1 using CPLEX 12.8 as LP solver + IPOPT 3.12.9 as nonlinear solver)
  - ✓ **basic**: our branch-and-cut algorithm <u>without</u> intersection cuts
  - ✓ **with-IC**: intersection cuts separated at each node where the LP solution is integral

- Single-thread runs (parallel runs not allowed in SCIP) with a time limit of **1 hour** on a standard PC Intel @ 3.10 GHz with 16 GB ram

- **Testbed**: all quadratic instances in **MINLPlib** (700+ instances) …
  … but some instances removed as root LP was **unbounded**
  → **620** instances left, **248** of which solved by all methods in 1 hour

Results

SCIP				basic				with-IC			
#opt	#fast	T_{ari}	T_{geo}	#opt	#fast	T_{ari}	T_{geo}	#opt	#fast	T_{ari}	T_{geo}
378	224	1480.58	34.45	328	156	1744.56	53.04	323	58	1787.33	67.93

Table 1: Results on 620 instances from the MINLPlib.

			SCIP			basic			with-IC		
Class	Time Range	#inst	#fast	T_{ari}	T_{geo}	#fast	T_{ari}	T_{geo}	#fast	T_{ari}	T_{geo}
G_1	(0,3600]	248	123	81.23	0.90	108	76.39	0.77	34	84.46	1.15
G_2	(1,3600]	118	37	170.65	8.19	62	160.48	6.05	22	177.42	10.67
G_3	(10,3600]	78	26	256.84	21.14	38	242.15	17.36	16	267.51	33.02
G_4	(100,3600]	41	20	464.30	34.33	15	449.58	87.02	6	489.46	105.46
G_5	(1000,3600]	12	5	1153.15	154.94	5	789.24	66.83	2	954.65	84.14

Table 3: Results on the 248 MINLPlib instances than can be solved by all methods within the 1-hour time limit.

More statistics

Class	Time Range	#inst	SCIP #fast	SCIP T_{ari}	SCIP T_{geo}	basic #fast	basic T_{ari}	basic T_{geo}	with-IC #fast	with-IC T_{ari}	with-IC T_{geo}
C_1	(0,1]	130	86	0.07	0.05	46	0.06	0.05	12	0.09	0.08
C_2	(1,10]	40	11	2.58	1.22	24	1.23	0.70	6	1.74	1.10
C_3	(10,100]	37	6	26.95	12.34	23	12.30	2.84	10	21.57	9.07
C_4	(100,1000]	29	15	179.26	18.38	10	309.03	97.06	4	296.96	115.79
C_5	(1000, 3600]	12	5	1153.15	154.94	5	789.24	66.83	2	954.65	84.14
ALL	(0,3600]	248	123	81.23	0.90	108	76.39	0.77	34	84.46	1.15

Table 2: Results on the 248 MINLPlib instances than can be solved by all methods within the 1-hour time limit.

ICs can make a difference

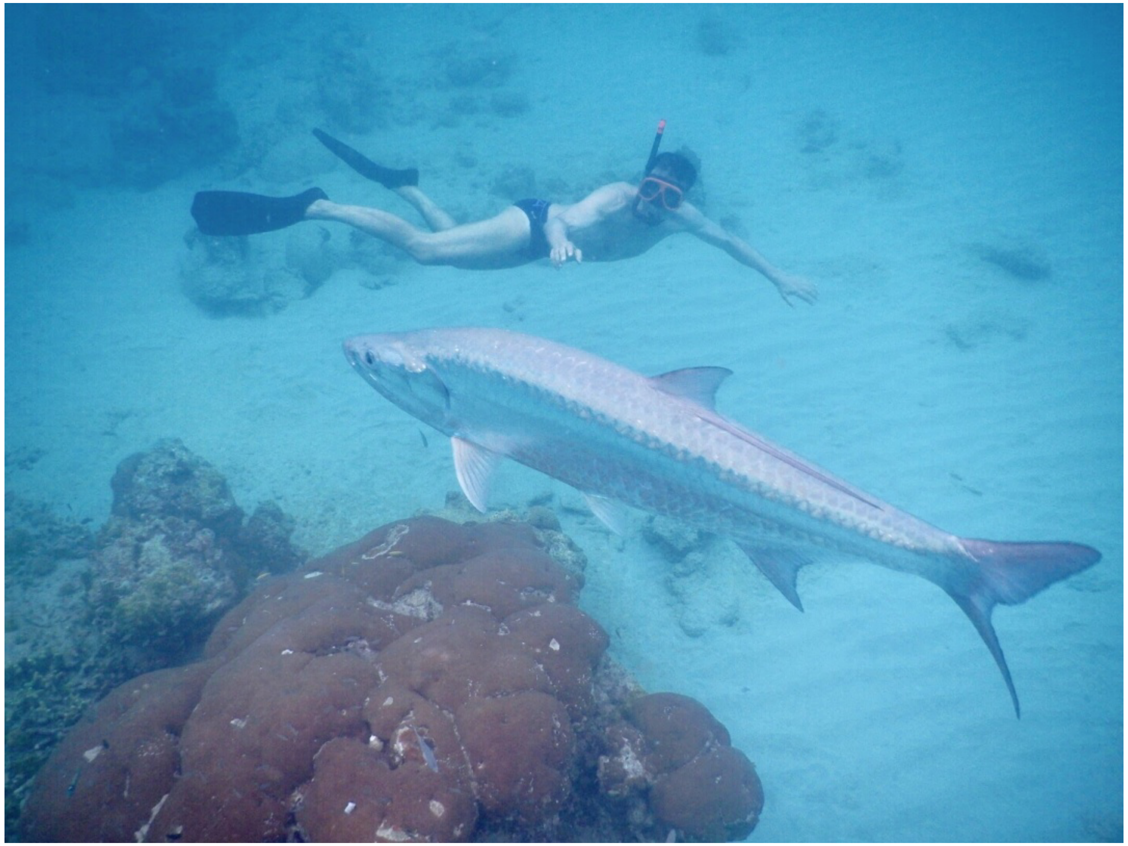
Instance	SCIP	basic	with-IC
blend531	234.21	3600.00	31.05
crudeoil_lee4_09	89.12	9.83	2.21
portfol_classical050_1	57.03	54.37	33.26
powerflow0009r	3600.00	3600.00	969.12
powerflow0014r	3600.00	3600.00	302.77
sporttournament14	3600.00	182.41	125.50
squfl015-080	3600.00	238.53	137.32
squfl025-030	3600.00	44.46	18.72
turkey	61.19	3600.00	0.11

Table 4: Selected instances for which adding intersection cuts is highly beneficial.

Thanks for your attention!

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

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



.