Mixed-Integer Nonconvex problems: an MILP perspective

Pietro Belotti

Lehigh University – September 11, 2008

Mixed-Integer Non-Linear Programming

$$
\begin{array}{llll}
\mathbf{P}_0) & \min & f(x) & \\
& s.t. & g_i(x) \leq 0 & i \in I \\
& & x_j^l \leq x_j \leq x_j^u & j \in N_0 = 1, 2 \ldots, n \\
& & x_j \in \mathbb{Z} & j \in J_0 \subseteq N_0
\end{array}
$$

- f and g_i's are, in general, nonconvex
- If f and g_i's are convex, we call $\mathbf{P}_0$ a **convex** MINLP
- f and g_i's are factorable: can be written as $\sum_{i=1}^{k} \prod_{j=1}^{p} h_{ij}(x)$, with h_{ij} univariate with factorable arguments

Applications

- Water treatment: Design of water networks with reuse of water, decentralized water treatment (minimize the consumption of fresh water)
- Scheduling and blending for production plants: coupling the problem of scheduling the production in a refinery and blending operations to get gasoline of different grades
- Trimloss problems for paper, wood, film, steel, glass industry – Mixed Integer bilinear problems (two sets of variables, formulation is linear in each set individually)
- Portfolio optimization. Convex in the classical case, but discrete if there are transaction (fixed) costs and nonconvex if robustness is introduced

Previous work

- Branch & Bound (B&B) (Gupta & Ravindran ’85; Tuy & Horst ’88; Nabar & Schrage ’91; Borchers & Mitchell ’94; Stubbs & Mehrotra ’99)
- Generalized Benders Decomposition (Geoffrion ’72)
- Outer-Approximation (Duran & Grossmann ’86; Yuan et al. ’88; Fletcher & Leyffer ’94)
- LP/NLP based B&B (Quesada & Grossmann ’92)

Software:

- Baron (Tawarmalani & Sahinidis)
- LaGO (Nowak & Vigerske)
- (convex) Bonmin (Bonami et al.), FilMINT (Abhishek, Leyffer, Linderoth)

How do we solve it?

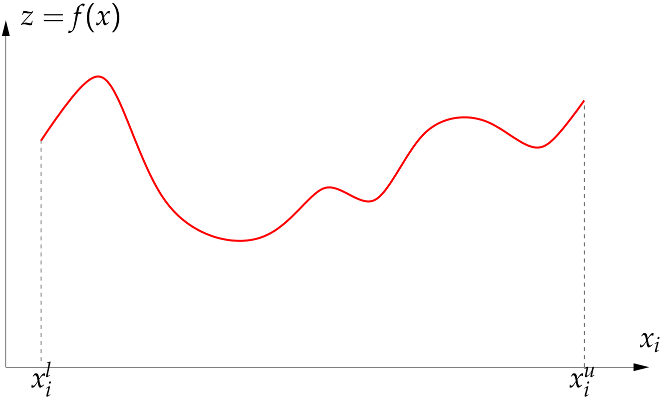
With a spatial Branch&Bound[1]: enumerate implicitly all local minima, use a convex (linear) relaxation to find lower bounds.

Key components:

- linearization (or convexification) for lower bounds
- heuristics for upper bounds
- branching rules to partition the solution set
- bound tightening to reduce the solution set

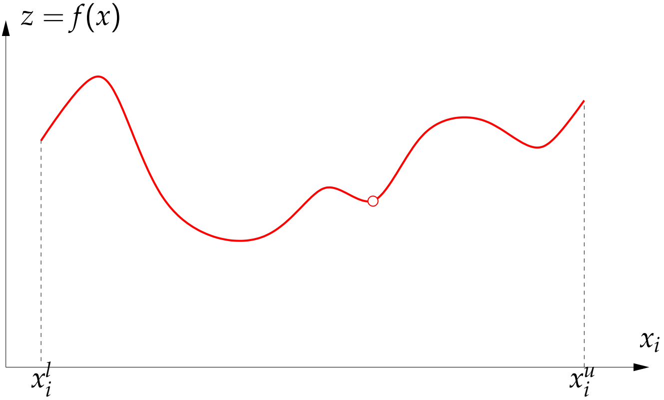
[1] See also Smith&Pantelides 1997, Tawarmalani&Sahinidis 2002

Solving nonconvex MINLPs

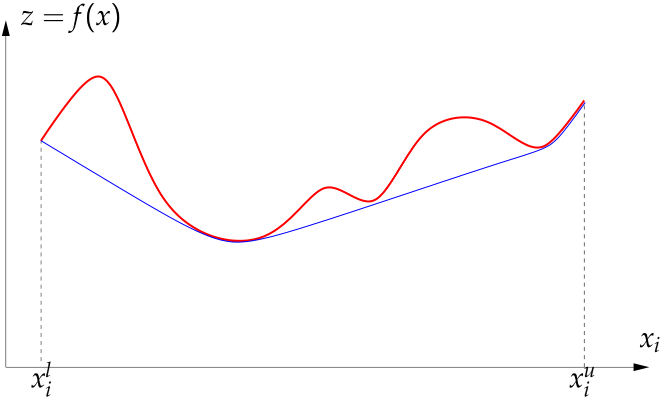


Relaxing integrality → nonconvex NLPs

Solving nonconvex MINLPs

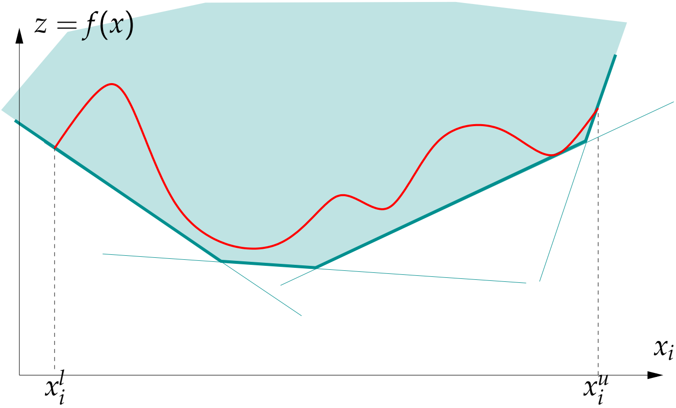


$\Rightarrow$ finding a valid lower bound is difficult (local minimum)



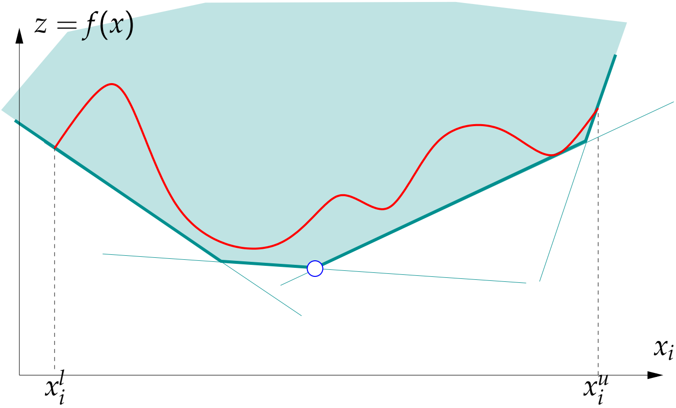
Usually, a convex relaxation is sought

Solving nonconvex MINLPs



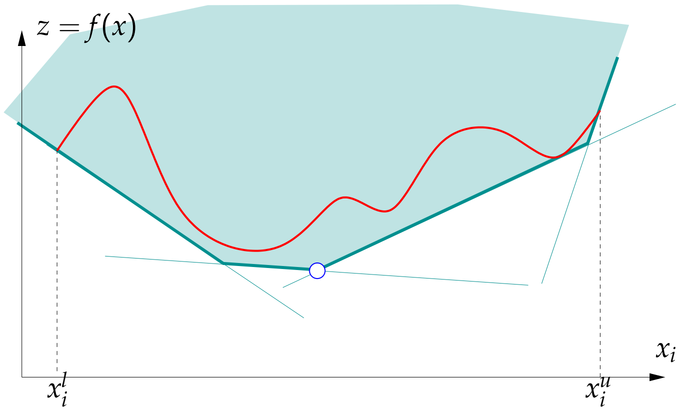
For instance, a **linear** relaxation

Solving nonconvex MINLPs



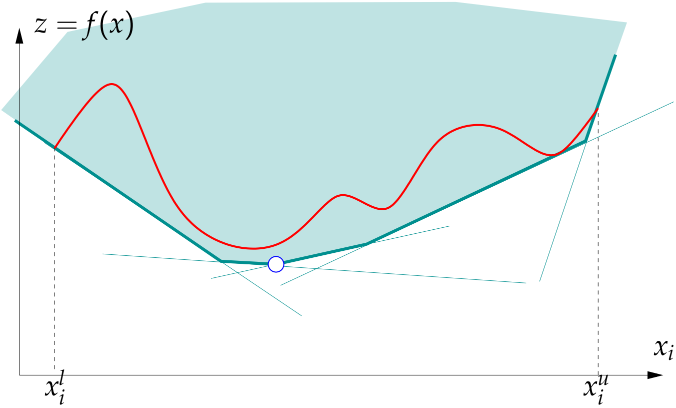
$\Rightarrow$ get a lower bound

Solving nonconvex MINLPs



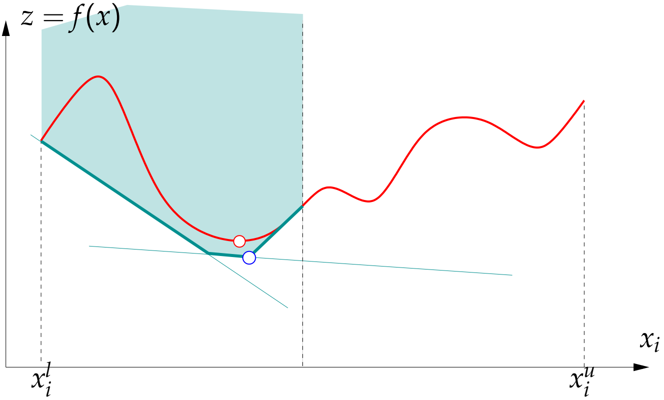
Solution may be NLP-infeasible (and/or fractional)

Solving nonconvex MINLPs



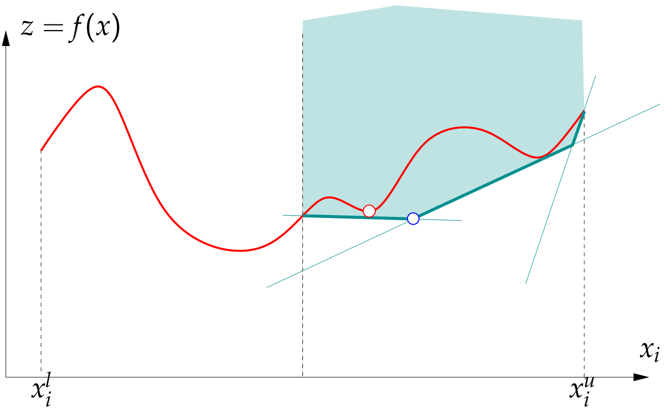
Either refine the linearization

Solving nonconvex MINLPs



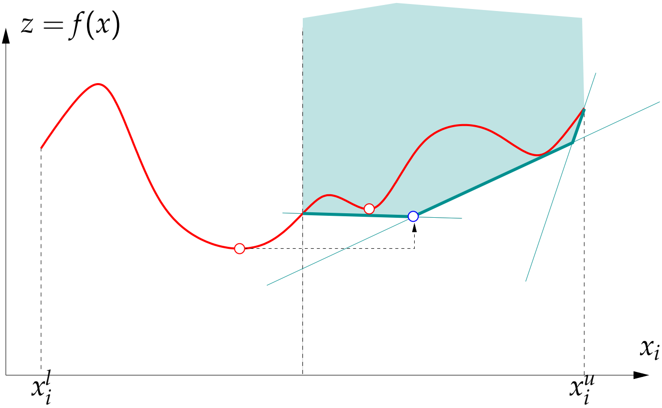
or branch on continuous variables

Solving nonconvex MINLPs



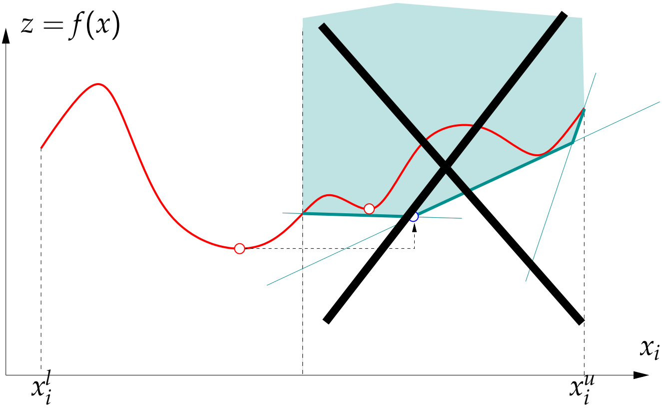
Linearization and lower bound improves

Solving nonconvex MINLPs



Linearization and lower bound improves

Solving nonconvex MINLPs



Linearization and lower bound improves

Couenne, a solver for nonconvex MINLPs

Couenne[2] is a Branch&Bound for nonconvex MINLPs. Written in C++, available as Open Source in Coin-OR (www.coin-or.org), it implements

- linearization of nonconvex functions
- heuristics for upper bound
- specialized branching rules
- bound tightening

It uses code from `Bonmin` (MINLP B&B), `Cbc` (Branch&Bound), `Cgl` (Cut generation), `Clp` (LP solver), `Ipopt` (NLP solver), and `LaGO` (quadratic forms).

---

[2] **C**onvex **O**ver/**U**nder **EN**velopes for **N**onlinear **E**stimation

A late outline

Already there

- linearization
- branching rules

Soon there

- disjunctive cuts
- nonconvex feasibility pump
- linearization cuts for MIQQP problems

Convex relaxations of non-convex MINLPs

$\mathbf{P}_0$ factorable $\Rightarrow$ can be reformulated (Smith&Pantelides, 1997)

- $\sum_{i=1}^k h_i(x)$ becomes $\boxed{\begin{array}{l}\sum_{i=1}^k x_{n+i} \\ x_{n+i} = h_i(x), 1! \leq i \leq k\end{array}}$
- $\prod_{i=1}^k h_i(x)$ becomes $\boxed{\begin{array}{l}\prod_{i=1}^k x_{n+i} \\ x_{n+i} = h_i(x), 1 \leq i \leq k\end{array}}$
- $h_1(h_2(x))$ becomes $h_1(x_2)$, with $x_2 = h_2(x)$
- $\ldots$

Recursively apply until all nonlinear constraints are of the form $x_k = \vartheta_k(x_1, x_2 \ldots, x_{k-1})$, with $\vartheta_k \in \Theta = \{\sum, \prod, \exp, \log, \sin, \text{abs} \ldots\}$.

Reformulation

The initial problem

$$
\begin{array}{llll}
\mathbf{P}_0) & \min & f(x) & \\
& s.t. & g_i(x) \leq 0 & i \in I \\
& & x_j^l \leq x_j \leq x_j^u & j \in N_0 = 1, 2 \ldots, n \\
& & x_j \in \mathbb{Z} & j \in J_0 \subseteq N_0
\end{array}
$$

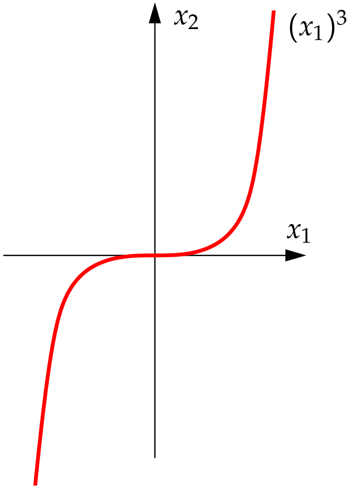
is **reformulated** as an equivalent problem

$$
\begin{array}{llll}
\mathbf{P}) & \min & x_{n+q} & \\
& s.t. & x_k = \vartheta_k(x_1, x_2 \ldots, x_{k-1}) & k = n+1, n+2 \ldots, n+q \\
& & x_j^l \leq x_j \leq x_j^u & j \in N = 1, 2 \ldots, n+q \\
& & x_j \in \mathbb{Z} & j \in J \subseteq N
\end{array}
$$

Then, each $x_k = \vartheta_k(x_1, x_2, \ldots, x_{k-1})$, $k = n+1, n+2 \ldots, n+q$, is linearized through inequalities $a_k x_k + A_k x \geq b_k$.

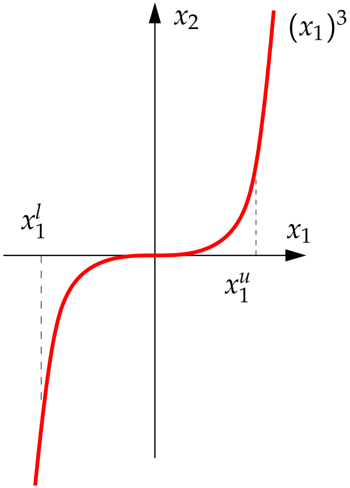
Linearization

$x_2 = \vartheta(x_1) = (x_1)^3$
with $x_1^l \leq x_1 \leq x_1^u$



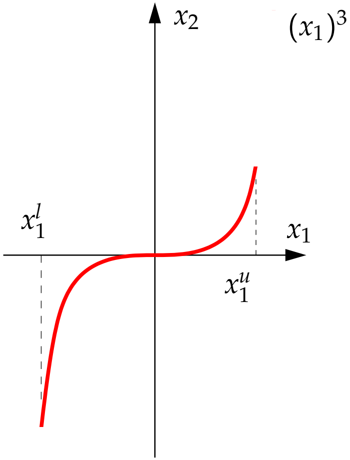
Linearization

$x_2 = \vartheta(x_1) = (x_1)^3$
with $x_1^l \leq x_1 \leq x_1^u$



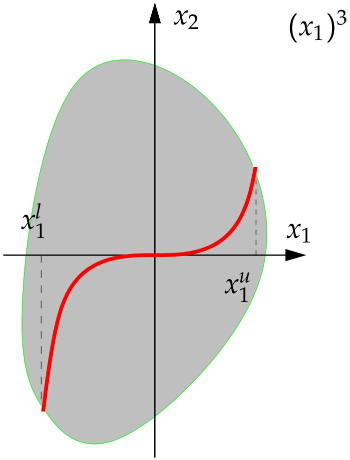
Linearization

$x_2 = \vartheta(x_1) = (x_1)^3$
with $x_1^l \leq x_1 \leq x_1^u$



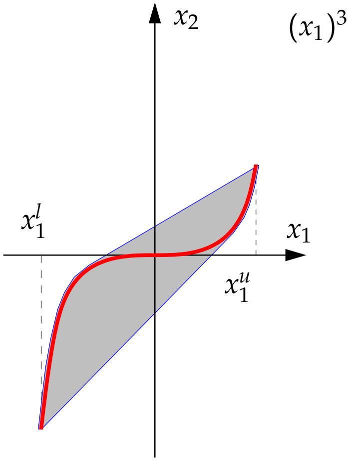
Linearization

$x_2 = \vartheta(x_1) = (x_1)^3$
with $x_1^l \le x_1 \le x_1^u$



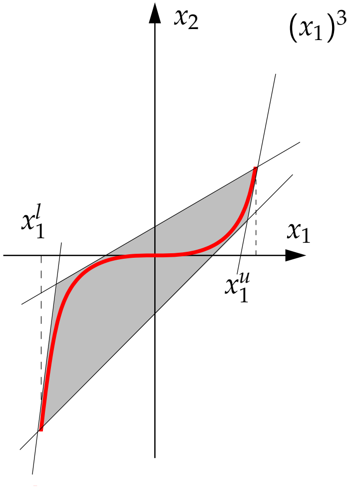
Linearization

$x_2 = \vartheta(x_1) = (x_1)^3$
with $x_1^l \leq x_1 \leq x_1^u$



Linearization

$x_2 = \vartheta(x_1) = (x_1)^3$
with $x_1^l \le x_1 \le x_1^u$



Linearization

Obtain the equivalent problem

$$
\begin{array}{llll}
\mathbf{P}) & \min & x_{n+q} & \\
& s.t. & x_k = \vartheta_k(x_1, x_2 \ldots, x_{k-1}) & k = n+1, n+2 \ldots, n+q \\
& & x_j^l \leq x_j \leq x_j^u & j \in N = 1, 2 \ldots, n+q \\
& & x_j \in \mathbb{Z} & j \in J \subseteq N
\end{array}
$$

Replace each $x_k = \vartheta_k(x_1, x_2, \ldots, x_{k-1})$, $k = n+1, n+2, \ldots, n+q$ with inequalities $a_k x_k + A_k x \geq b_k$.

$$
\begin{array}{llll}
\mathbf{LP}) & \min & x_{n+q} & \\
& s.t. & a_k x_k + A_k x \geq b_k & k = n+1, n+2 \ldots, n+q \\
& & x_j^l \leq x_j \leq x_j^u & j \in N = 1, 2 \ldots, n+q \\
& & x_j \in \mathbb{Z} & j \in J \subseteq N
\end{array}
$$

A linear relaxation providing a valid lower bound.

Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

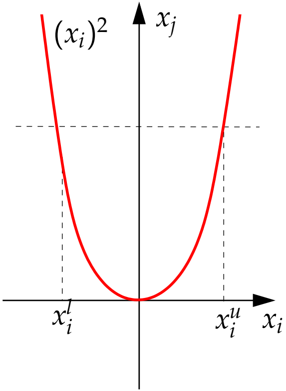
(⋆) add linearization cuts

if no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

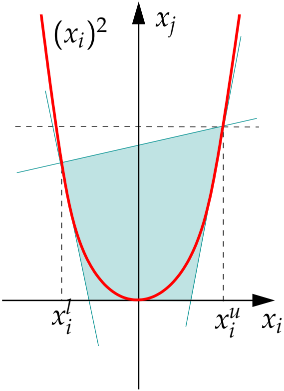
▶ ($\star$) add linearization cuts

if no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

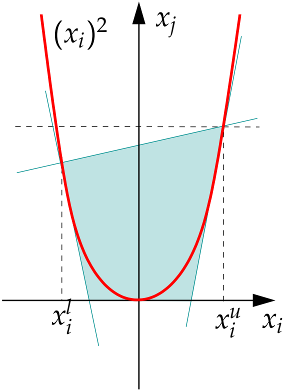
(⋆) add linearization cuts

▶ **if** no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

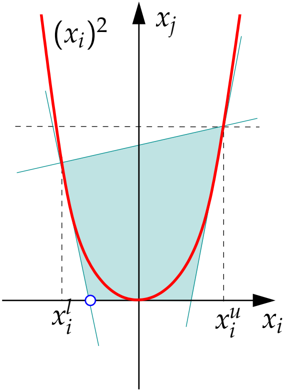
(⋆) add linearization cuts

if no cuts found, **break**

▶ get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

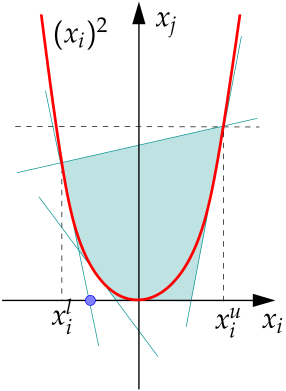
(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

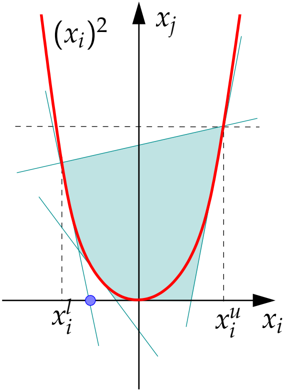
- (Lb) **repeat** k times:
  - ▶ ($\star$) add linearization cuts
  - **if** no cuts found, **break**
  - get *lower bd.*, soln. $\hat{x}$
- (Ub) Look for a *feasible solution* with NLP solver
- (Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

- (Lb) **repeat** k times:
  - ($\star$) add linearization cuts
  - ▶ **if** no cuts found, **break**
  - get *lower bd.*, soln. $\hat{x}$
- (Ub) Look for a *feasible solution* with NLP solver
- (Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

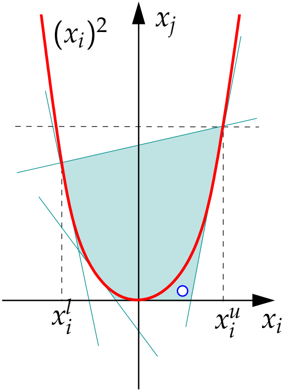
($\star$) add linearization cuts

if no cuts found, **break**

▶ get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

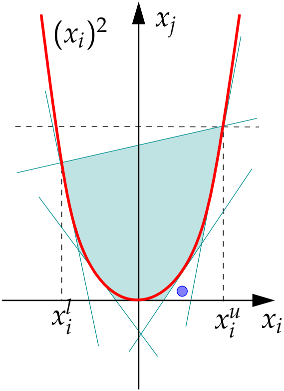
▶ ($\star$) add linearization cuts

if no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

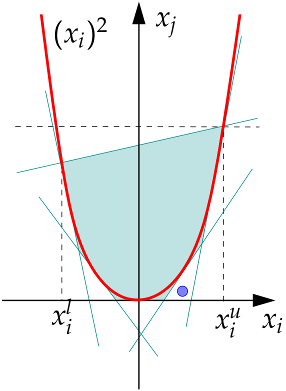
($\star$) add linearization cuts

▶ **if** no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

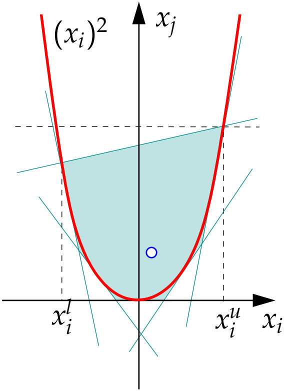
(⋆) add linearization cuts

if no cuts found, **break**

▶ get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

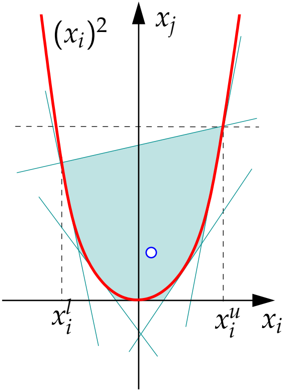
► ($\star$) add linearization cuts

if no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

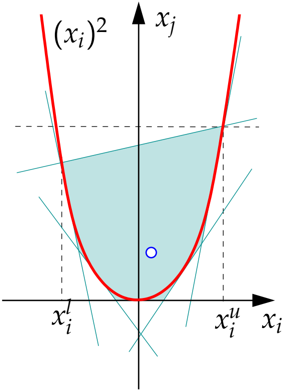
(⋆) add linearization cuts

▶ **if** no cuts found, **break**

get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

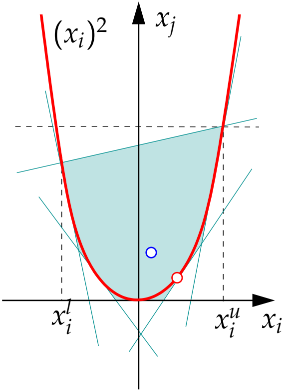
  ($\star$) add linearization cuts

  if no cuts found, **break**

  get *lower bd.*, soln. $\hat{x}$

▶(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

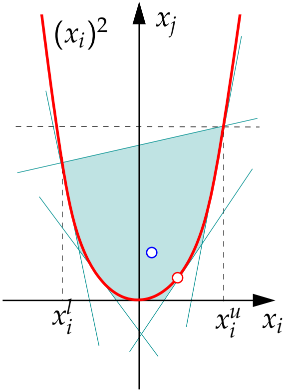
  ($\star$) add linearization cuts

  if no cuts found, **break**

  get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

▶(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

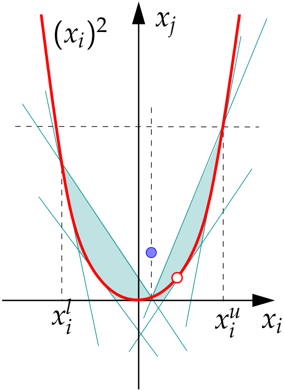
  ($\star$) add linearization cuts

  if no cuts found, **break**

  get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

▶(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



Spatial B&B with convexification

At each node:

(Lb) **repeat** k times:

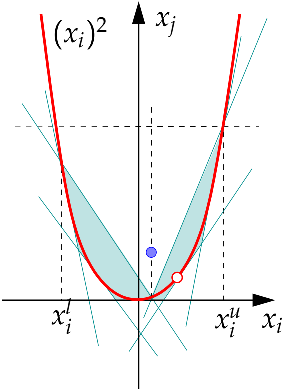
  ($\star$) add linearization cuts

  if no cuts found, **break**

  get *lower bd.*, soln. $\hat{x}$

(Ub) Look for a *feasible solution* with NLP solver

(Br.) If $x_k = \theta_k(x)$ *infeasible*, i.e. $\hat{x}_k \neq \theta_k(\hat{x})$, branch on x



($\star$) solves a **separation** problem: separate current iterate from **convex envelope of** $\{x \in \mathbb{R}^{n+q} : x_k = \theta_k(x)\}$

Branching rules

Purpose: partition the solution set in $\geq$ two (easier) subproblems.

- MILP: if a component $x_i^\star$ of the LP solution is fractional, create subproblems: P_1 with branching rule $x_i \leq \lfloor x_i^\star \rfloor$; P_2 with $x_i \geq \lceil x_i^\star \rceil$
- MINLP: may be necessary for continuous variables.

Branching rules

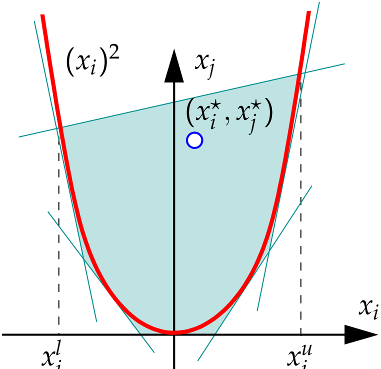
Purpose: partition the solution set in $\geq$ two (easier) subproblems.

- MILP: if a component $x_i^\star$ of the LP solution is fractional, create subproblems: P_1 with branching rule $x_i \leq \lfloor x_i^\star \rfloor$; P_2 with $x_i \geq \lceil x_i^\star \rceil$
- MINLP: may be necessary for continuous variables.

Branching rules

Purpose: partition the solution set in $\geq$ two (easier) subproblems.

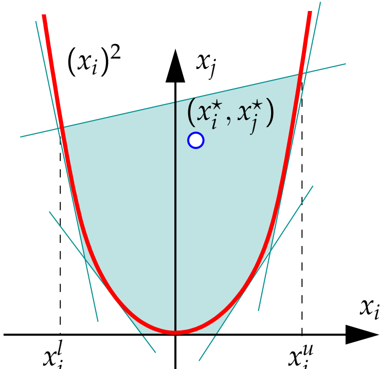
- MILP: if a component $x_i^\star$ of the LP solution is fractional, create subproblems: P_1 with branching rule $x_i \leq \lfloor x_i^\star \rfloor$; P_2 with $x_i \geq \lceil x_i^\star \rceil$
- MINLP: may be necessary for continuous variables.



Branching rules

Purpose: partition the solution set in $\geq$ two (easier) subproblems.

- MILP: if a component $x_i^\star$ of the LP solution is fractional, create subproblems: P_1 with branching rule $x_i \leq \lfloor x_i^\star \rfloor$; P_2 with $x_i \geq \lceil x_i^\star \rceil$
- MINLP: may be necessary for continuous variables.



Strong branching

Strong branching[3]: for each branching candidate x_i,

- simulate br. rule $x_i \leq x_i^b$, re-solve $\to$ new lower bound $x_{n+q}^{\leq}$
- simulate br. rule $x_i \geq x_i^b$, re-solve $\to$ new lower bound $x_{n+q}^{\geq}$
- set $U_i^{\text{strong}} = \alpha \min\{x_{n+q}^{\leq}, x_{n+q}^{\geq}\} + (1-\alpha)\max\{x_{n+q}^{\leq}, x_{n+q}^{\geq}\}$, with $0 < \alpha < 1$

Choose variable x_i with maximum U_i^{strong}

Computationally expensive...
$\Rightarrow$Pseudocosts[4], Reliability Branching[5]: statistics on U_i^{strong} at initial nodes are used to *estimate* it at later nodes

[3] Applegate et al., "The TSP, a computational study".
[4] Benichou et al, "Experiments in MIP", *MathProg* '71
[5] Achterberg et al., "Branching rules revisited", *OR letters* 2005

Comparing with Baron

Baron (Branch And Reduce Optimization Navigator) is currently the state-of-the-art MINLP solver (Tawarmalani&Sahinidis 2002).

- A spatial branch&bound with linearization, bound reduction, and heuristics
- Uses external LP and NLP solvers (Cplex 9 and Minos)
- Couenne with reliability branching

Comparing with Baron – MINLP problems

Name	#var	#int	#con	Couenne time (lb)	Couenne ub	Baron time (lb)	Baron ub
non-convex MINLP							
Multistage	185	39	265	(-17621.4)	–	70.88	-7581
barton-aiche1	818	66	987	(-102.47)	–	(-103.31)	-81.8659
c-sched-4-7	233	168	138	(-254146)	–	(-1.93e+05)	-1.33e+05
ex1233	48	12	52	(76225.9)	161022	169.80	1.55e+05
ex1243	57	29	75	4.95	83402.5	1.33	83402.5
ex1244	86	40	110	(68674.3)	85431.1	25.01	82040.0
ex1252	39	15	43	124.50	128894	0.23	128894
nous1	48	2	41	(1.510)	1.567	169.20	1.567
nous2	48	2	41	277.26	0.626	1.22	0.626
nConvPl	948	148	920	(-8790.4)	-4580	(-8946.17)	-7529.3
space-25	893	750	235	(84.61)	–	(155.566)	784.84
space-25-r	818	750	160	(71.72)	–	(160.507)	786.34
feedloc	89	96	247	114.54	0	2.93	0
MIQQP[6]							
ibell3a	122	209	104	1164.90	878785	(-3.36e+09)	2966916.97
ibienst1	505	83	576	4065.70	48.74	(-2.42e+09)	48.74
imisc07	260	957	212	(2501.63)	2814.28	–	–
iran8x32	512	767	296	4643.10	5255.45	–	–
conic (convex) MINLP[7]							
classical_40_0	120	41	83	1233.30	-0.0815	218.56	-0.0815
classical_40_1	120	41	83	98.04	-0.0847	20.75	-0.0847
robust_20_0	83	22	66	4.42	-0.0798	2.60	-0.0798
robust_20_1	83	22	66	25.85	-0.0533	16.39	-0.0533
shortfall_20_0	84	22	67	56.89	-1.090	6.90	-1.090
shortfall_20_1	84	22	67	(-1.076)	-1.066	35.79	-1.075

[6] H. Mittelmann, `http://plato.asu.edu/ftp/miqp`

[7] Vielma, Ahmed, & Nemhauser 2008

Comparing with Baron – pure NLP problems

			Couenne		Baron	
Name	#var	#con	time (lb)	ub	time (lb)	ub
Hicks_5	83	68	7.37	227.26	21.89	227.26
Hicks_20	338	278	110.45	227.26	334.40	229.7
Hicks_50	848	698	755.10	227.26	3968	227.26
Hicks_100	1698	1398	3994.40	227.26	–	–
ex5_2_5	33	19	(-7211.96)	-3500	(-5055)	-3500
ex5_3_3	62	53	(1.89745)	3.234	(2.174)	3.234
foulds3	168	48	(-59.7432)	-8	(-69.809)	-8
QCQP						
dualc8	9	16	(18309.0)	18309.2	(18306.3)	18309.1
dual1	86	2	(-205.22)	0.035	(-176.46)	0.035
dual4	76	2	–	–	(-198.742)	0.746
qadlittl	97	54	3775.80	480319	216.93	480319
qp1	50	2	(-0.0831)	8.093e-4	(-0.304)	8.093e-4
qp2	50	2	(-0.0891)	8.093e-4	(-0.305)	8.093e-4
qp3	100	52	(-0.2905)	8.093e-4	(-0.093)	8.093e-4
cvxqp1_s	101	51	(10767.4)	12467.9	(9739.53)	11590.7
cvxqp2_s	101	26	(7298.61)	–	(6828.31)	8120.94
cvxqp3_s	101	76	(11943)	–	166.25	11943.4
primal4	1490	76	1650.50	-0.746	(-0.779)	0
qetamacr	543	334	2194.80	86760.4	(61835.2)	86760.4
gouldqp2	700	350	(-0.165)	1.84e-4	(-0.186)	1.84e-4
qisrael	143	164	686.72	2.5347e+7	78.92	2.53e+07
qshare1b	221	111	(720058)	–	983.35	720078
stcqp1	3159	1	1598.70	155144	(148327.34)	157758.85
values	203	2	840.27	-1.39	(-12.90)	-1.39

Comparing with Baron – box QP problems

		Couenne		Baron	
Name	#var	time (lb)	ub	time (lb)	ub
spar030-060-1	30	2081.10	-706	(-830.759)	-706
spar030-060-2	30	24.43	-1377.17	3.62	-1377.17
spar040-050-1	40	(-1188.02)	-1154.5	(-1403.72)	-1154.5
spar040-050-2	40	4053.90	-1430.98	(-1636.44)	-1430.98
spar040-060-1	60	(-1718.03)	-1311.06	(-2009.22)	-1322.67
spar050-040-1	60	(-1597.44)	-1411	(-1900.97)	-1411
spar050-050-1	70	(-2203.13)	-1193	(-2685.19)	-1198.41
spar060-020-1	80	470.49	-1212	3582.38	-1212
spar060-020-2	90	45.11	-1925.5	55.06	-1925.5
spar070-025-1	70	(-2856.69)	-2538.91	(-3169.19)	-2538.91
spar080-025-1	80	(-3758.34)	-3157	(-4173.78)	-3157
spar090-025-1	90	(-4861.59)	-3361.5	(-5468.25)	-3372.5

Improving Couenne

As Open Source code, **Couenne** can be used as a *base* for the development and test of MINLP models/solvers, by

- specializing linearization, branching, heuristic, and bound reduction techniques
- extending the set of operators: quadratic forms, polynomials…
- generalizing MILP techniques to MINLP

Augmenting the set of operators

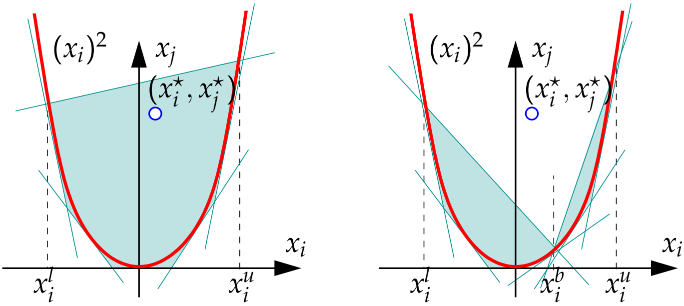
- any function $f : \mathbb{R}^n \to \mathbb{R}$ can do
- must provide procedure to **separate** a point $x \in \mathbb{R}^{n+1}$ from

$$\mathrm{convenv}\{x \in \mathbb{R}^{n+1} : x_{n+1} = f(x_1, x_2 \ldots, x_n)\}$$

```
class polydeg4:  public expression {

    [...]

    double compute(double x) {
        return myComputePoly4 (x);
    }

    void generateCuts(double *curpoint) {
        // separation procedure
    }
}
```

MILP extension #1: Disjunctive cuts

- Disjunctions arise naturally in Integer Programming and also in nonconvex MINLP!
- Recently used in non-MILP contexts – e.g. Saxena et al., IPCO 2008, for MIQQPs: disjunction from $X - xx^T \preceq 0$
- In nonconvex (MI)NLP, disjunction are provided by branching rules on nonlinear expressions.



Disjunctive cuts for nonconvex MINLP

Consider a nonconvex MINLP and its current linearization, $\min\{x_{n+q} : Ax \leq a, l \leq x \leq u\}$. Optimal LP solution is $x^\star$.

- branching rule on $x_i \Rightarrow$ two *refined* linearizations:
  - $x_i \leq b$: extra inequalities $Bx \leq b$
  - $x_i \geq b$: extra inequalities $Dx \leq d$
- create the *Cut Generating LP*, find deepest cut $\alpha x \leq \beta$:

$$
\begin{array}{llllllll}
\max & \alpha x^\star & -\beta & & & & & \\
 & \alpha & & = uA & & +u'B & & \\
 & \alpha & & = & vA & & +v'D \\
 & & \beta & \geq ua & & +u'b & & \\
 & & \beta & \geq & va & & +v'd \\
 & & & \multicolumn{5}{l}{||(u, v, u', v')||_1 = 1}
\end{array}
$$

- cons: one huge LP for just one cut?

MILP extension #2: Nonconvex Feasibility Pump

- principle: two sequences of points
  - $\hat{x}^k$ Integer but infeasible for the relaxation
  - $\bar{x}^k$ Fractional but feasible for the relaxation
- Originally introduced for MILP (Fischetti, Glover, Lodi)
- Extended to convex MINLP (Bonami, Cornuéjols, Lodi, Margot)

MILP extension #2: Nonconvex Feasibility Pump

FP for MILP (Fischetti et al.):
$\min\{c^T x : Ax \geq b, x_i \in \mathbb{Z} \forall i \in J \subseteq N\}$

- $\bar{x}^0 = \mathrm{argmin}\{c^T x : Ax \geq b\}$; let $k \leftarrow 0$
- **while** $\bar{x}^k$ not integer

  let $\hat{x}^k := \lfloor \bar{x}^k \rceil$

  let $\bar{x}^{k+1} := \mathrm{argmin}\{||x - \hat{x}^k||_1 : Ax \geq b\}$

  let $k \leftarrow k+1$

MILP extension #2: Nonconvex Feasibility Pump

FP for convex MINLP (Bonami et al.):
$\min\{f(x) : g(x) \leq 0, x_i \in \mathbb{Z} \forall i \in J \subseteq N\}$

- $\bar{x}^0 = \text{argmin}\{f(x) : g(x) \leq 0\}$; let $k \leftarrow 0$
- **while** $\bar{x}^k$ not integer

  let $\hat{x}^k := \text{argmin}\{||x - \bar{x}^k||_1 : Ax \geq b, x_i \in \mathbb{Z} \forall i \in J \subseteq N\}$

  let $\bar{x}^{k+1} := \text{argmin}\{||x - \hat{x}^k||_1 : g(x) \leq 0\}$

  let $k \leftarrow k + 1$

[$Ax \geq b$ is an **Outer Approximation** of the problem]

FP extends naturally to nonconvex MINLP, if Outer Approximation is replaced by linearization inequalities.

Current work #3: linearization of MIQQP

Joint work with F. Margot, A. Qualizza (CMU)

- consider the MIQQP: $\min\{x^T Q_0 x + a_0^T x : x^T Q_i x + a_i^T x \leq b_i\}$, Q_i not PSD (in general)
- reformulate: $\min\{Q_0 \bullet X + a_0^T x : Q_i \bullet X + a_i^T x \leq b_i, X = xx^T\}$
- $X = xx^T$ means $X - xx^T \succeq 0$ and $X - xx^T \preceq 0$
- $X - xx^T \succeq 0$ equals $\begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} = \tilde{X} \succeq 0$
- Linearize $\tilde{X} \succeq 0$ by separating[8] cuts of the form $a^T \tilde{X} a \geq 0$ for any vector a

[8] See also Sherali & Fraticelli, Sivaramakhrishnan & Mitchell…

Future work

Interface to generate problems at code level (analogous to Ilog's Concert Technology or GLPK C++ interface)

```
int main (int argc, char **argv) {

    CouenneVar x1, x2;

    CouenneConstraint c1 = x1^2 + x2^2 <= 1;
    CouenneMinObj o1 = 2*x1 + 3*x2;

    CouenneProblem p;
    p << x1 << x2 << o1 << c1;
    p.solve();

    CouenneConstraint c2 = x2 >= .5;
    p << c2;
    p.resolve();
}
```

Resources

- P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, "Branching and bounds tightening techniques for non-convex MINLP," `opt-online` (Sep. 2008).

http://egon.cheme.cmu.edu/ibm/page.htm

https://projects.coin-or.org/Bonmin/wiki/BonCouenne

(or google "boncouenne")