Mixed Integer Non Linear Optimization: Methods and Applications
–
Mixed Integer Nonlinear Programming

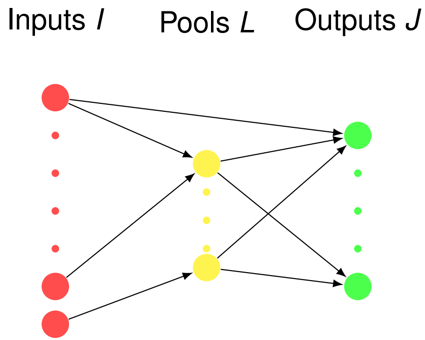
Claudia D'Ambrosio
dambrosio@lix.polytechnique.fr

ÉCOLE POLYTECHNIQUE
IP PARIS

Outline

Pooling Problem

Pooling Problem



- Nodes $N = I \cup L \cup J$
- Arcs A
  $(i,j) \in (I \times L) \cup (L \times J) \cup (I \times J)$
  on which materials flow
- Material attributes: K
- Arc capacities: u_{ij}, $(i,j) \in A$
- Node capacities: C_i, $i \in N$
- Attribute requirements
  $\alpha_{kj}, \; k \in K, j \in J$

Pooling Problem: Motivation

- refinery processes in the **petroleum industry**

Pooling Problem: Motivation

- refinery processes in the **petroleum industry**
- different **specifications**: e.g., sulphur/carbon concentrations or physical properties such as density, octane number, **...**

Pooling Problem: Motivation

- refinery processes in the **petroleum industry**
- different **specifications**: e.g., sulphur/carbon concentrations or physical properties such as density, octane number, ...
- wastewater treatment, e.g., Karuppiah and Grossmann (2006)

Pooling Problem: Motivation

- refinery processes in the **petroleum industry**
- different **specifications**: e.g., sulphur/carbon concentrations or physical properties such as density, octane number, **...**
- wastewater treatment, e.g., Karuppiah and Grossmann (2006)
- Formally introduced by **Haverly (1978)**

Pooling Problem: Motivation

- refinery processes in the **petroleum industry**
- different **specifications**: e.g., sulphur/carbon concentrations or physical properties such as density, octane number, **...**
- wastewater treatment, e.g., Karuppiah and Grossmann (2006)
- Formally introduced by **Haverly (1978)**
- Alfaki and Haugland (2012) formally proved it is strongly **NP-hard**

Pooling problem: Citations

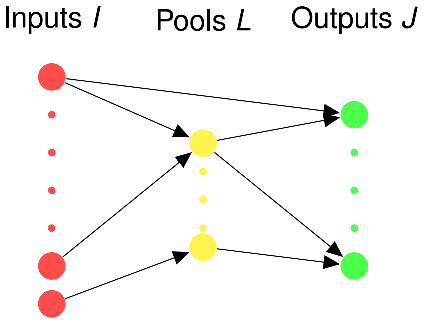
- Haverly, *Studies of the behaviour of recursion for the pooling problem*, ACM SIGMAP Bulletin, 1978
- Adhya, Tawarmalani, Sahinidis, *A Lagrangian approach to the pooling problem*, Ind. Eng. Chem., 1999
- Audet et al., *Pooling Problem: Alternate Formulations and Solution Methods*, Manag. Sci., 2004
- Liberti, Pantelides, *An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms*, JOGO, 2006
- Misener, Floudas, *Advances for the pooling problem: modeling, global optimization, and computational studies*, Appl. Comput. Math., 2009
- D'Ambrosio, Linderoth, Luedtke, *Valid inequalities for the pooling problem with binary variables*, IPCO, 2011
- Tawarmalani and Sahinidis. Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications, Ch. 9. Kluwer Academic Publishers, 2002.

"Simple" constraints

Variables x_{ij} for flow on arcs

Flow balance constraints at pools:

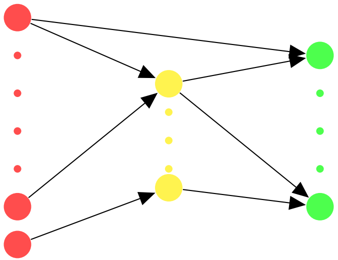
$$\sum_{i \in I_l} x_{il} - \sum_{j \in J_l} x_{lj} = 0, \quad \forall l \in L$$



Example: Pooling Problem

"Simple" constraints

Variables x_{ij} for flow on arcs



Flow balance constraints at pools:

$$\sum_{i \in I_l} x_{il} - \sum_{j \in J_l} x_{lj} = 0, \quad \forall l \in L$$

Capacity constraints:

$$\sum_{j \in J_i} x_{ij} + \sum_{l \in L_i} x_{il} \leq C_i, \quad \forall i \in I$$

$$\sum_{j \in J_l} x_{lj} \leq C_l, \quad \forall l \in L$$

$$\sum x_{ij} + \sum x_{lj} \leq C_j, \quad \forall j \in J$$

Example: Pooling Problem

"Complicating" constraints

- Inputs have associated attribute concentrations $\lambda_{ki},\ k \in K, i \in I$
- Concentration of attribute in pool is the weighted average of the concentrations of its inputs.
- This results in bilinear constraints.

Example: Pooling Problem

"Complicating" constraints

- Inputs have associated attribute concentrations $\lambda_{ki},\ k \in K, i \in I$
- Concentration of attribute in pool is the weighted average of the concentrations of its inputs.
- This results in bilinear constraints.

- **P-formulation** (Haverly 78):
  Keep track of concentration p_{kl} of attribute k in pool l

Example: Pooling Problem

"Complicating" constraints

- Inputs have associated attribute concentrations $\lambda_{ki},\ k \in K, i \in I$
- Concentration of attribute in pool is the weighted average of the concentrations of its inputs.
- This results in bilinear constraints.

- **P-formulation** (Haverly 78):
  Keep track of concentration p_{kl} of attribute k in pool l
- **Q-formulation** (Ben-Tal et al. 94):
  Variables q_{il} for proportion of flow into pool l coming from input i

Example: Pooling Problem

P-formulation

$$\sum_{j \in J_i} x_{ij} + \sum_{l \in L_i} x_{il} \leq C_i, \qquad \forall i \in I$$

$$\sum_{j \in J_l} x_{lj} \leq C_l, \qquad \forall l \in L$$

$$\sum_{i \in I_j} x_{ij} + \sum_{l \in L_j} x_{lj} \leq C_j, \qquad \forall j \in J$$

$$\sum_{i \in I_l} x_{il} - \sum_{j \in J_l} x_{lj} = 0, \qquad \forall l \in L$$

$$p_{kl} = \frac{\sum_{i \in I_l} \lambda_{ki} x_{il}}{\sum_{j \in J_l} x_{lj}} \qquad \forall k \in K, l \in L$$

$$\frac{\sum_{i \in I_j} \lambda_{ki} x_{ij} + \sum_{l \in L_j} p_{kl} x_{lj}}{\sum_{i \in I_j \cup L_j} x_{ij}} \leq \alpha_{kj}, \qquad \forall k \in K, j \in J$$

Example: Pooling Problem

P-formulation

$$\sum_{j\in J_i} x_{ij} + \sum_{l\in L_i} x_{il} \leq C_i, \qquad \forall i \in I$$

$$\sum_{j\in J_l} x_{lj} \leq C_l, \qquad \forall l \in L$$

$$\sum_{i\in I_j} x_{ij} + \sum_{l\in L_j} x_{lj} \leq C_j, \qquad \forall j \in J$$

$$\sum_{i\in I_l} x_{il} - \sum_{j\in J_l} x_{lj} = 0, \qquad \forall l \in L$$

$$\mathbf{p_{kl}} \sum_{\mathbf{j\in J_l}} \mathbf{x_{lj}} = \sum_{i\in I_l} \lambda_{ki} x_{il} \qquad \forall k \in K, l \in L$$

$$\sum_{i\in I_j} \lambda_{ki} x_{ij} + \sum_{l\in L_j} \mathbf{p_{kl} x_{lj}} \leq \alpha_{kj} \sum_{i\in I_j \cup L_j} x_{ij}, \qquad \forall k \in K, j \in J$$

Example: Pooling Problem

Q-formulation

$$x_{il} = q_{il} \sum_{j \in J_l} x_{lj}, \quad \forall i \in I, l \in L_i$$

$$\sum_{i \in I_l} q_{il} = 1, \quad \forall l \in L$$

Example: Pooling Problem

Q-formulation

$$x_{il} = q_{il} \sum_{j \in J_l} x_{lj}, \quad \forall i \in I, l \in L_i$$

$$\sum_{i \in I_l} q_{il} = 1, \qquad \forall l \in L$$

- Attribute constraints

$$\sum_{i \in I_j} \lambda_{ki} x_{ij} + \sum_{l \in L_j} x_{lj} \left(\sum_{i \in I_l} \lambda_{ki} q_{il} \right) \leq \alpha_{kj} \sum_{i \in I_j \cup L_j} x_{ij}, \quad \forall k \in K, j \in J$$

Example: Pooling Problem

Q-formulation

$$\sum_{j \in J_i} x_{ij} + \sum_{l \in L_i} x_{il} \leq C_i, \qquad \forall i \in I$$

$$\sum_{j \in J_l} x_{lj} \leq C_l, \qquad \forall l \in L$$

$$\sum_{i \in I_j} x_{ij} + \sum_{l \in L_j} x_{lj} \leq C_j, \qquad \forall j \in J$$

$$x_{il} - \mathbf{q_{il}} \sum_{\mathbf{j \in J_l}} \mathbf{x_{lj}} = 0 \qquad \forall i \in I, l \in L_i$$

$$\sum_{i \in I_l} q_{il} = 1 \qquad \forall l \in L$$

$$\sum_{i \in I_j} \lambda_{ki} x_{ij} + \sum_{l \in L_j} \mathbf{x_{lj}} \left(\sum_{i \in I_l} \lambda_{ki} \mathbf{q_{il}} \right) \leq \alpha_{kj} \sum_{i \in I_j \cup L_j} x_{ij}, \quad \forall k \in K, j \in J$$

From NLP to MINLP

- Decide whether to install pipes or not (0/1 decision)
- Associate a binary variable z_{ij} with each pipe (suppose for now on arcs from input to output)

Extra constraints:

$$\begin{aligned} x_{ij} &\leq \min(C_i, C_j) z_{ij} && \forall i \in I, j \in J_i \\ z_{ij} &\in \{0, 1\} && \forall i \in I, j \in J_i \end{aligned}$$

Objective Function

- Fixed cost for installing pipe

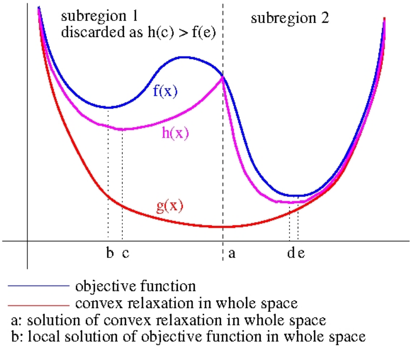
$$\min \sum_{i \in I} c_i \left(\sum_{l \in L_i} x_{il} + \sum_{j \in J_i} x_{ij} \right) - \sum_{j \in J} p_j \left(\sum_{i \in I_j} x_{ij} + \sum_{l \in L_j} x_{lj} \right) + \sum_{i \in I} \sum_{j \in J_i} f_{ij} z_{ij}$$

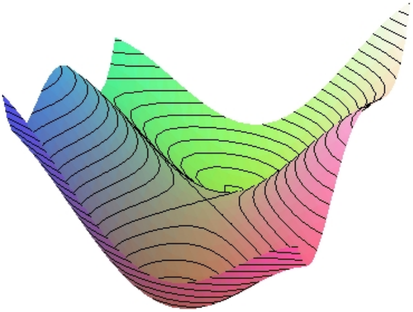
Outline

1. Motivating Applications

2. Global Optimization methods
   - Multistart
   - Spatial Branch-and-Bound
     - Standard form
     - Convexification
     - Expression trees
     - Variable ranges
     - Bounds tightening
     - Reformulation Linearization Technique (RLT)

Global Optimization methods





Exact

- "Exact" in continuous space: ε-approximate (*find solution within pre-determined* ε *distance from optimum in obj. fun. value*)
- For some problems, finite convergence to optimum ($\varepsilon = 0$)

Heuristic

- Find solution with probability 1 in infinite time

Outline

Multistart

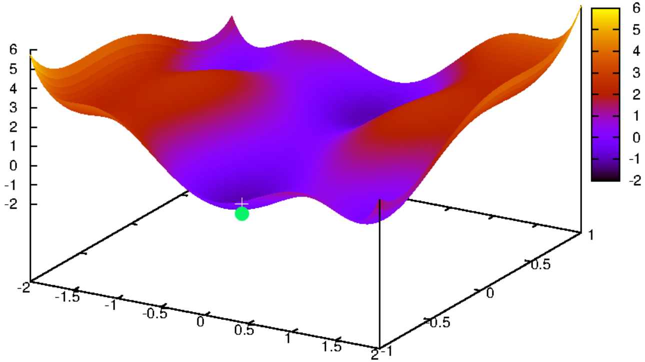
- The easiest GO method

$f^* = \infty$
$x^* = (\infty, \ldots, \infty)$
while $\neg$ termination **do**
    $x' = (\text{random}(), \ldots, \text{random}())$
    $x = \text{localSolve}(P, x')$
    if $f_P(x) < f^*$ **then**
        $f^* \leftarrow f_P(x)$
        $x^* \leftarrow x$
    end if
end while

- Termination condition: e.g. *repeat k times*

Six-hump camelback function

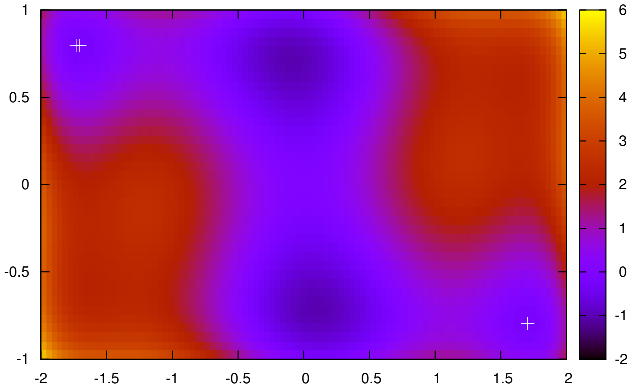
$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$



Global optimum (COUENNE)

Six-hump camelback function

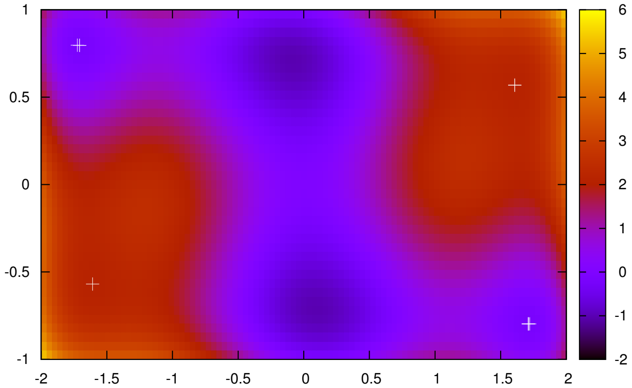
$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$



Multistart with IPOPT, $k = 5$

Six-hump camelback function

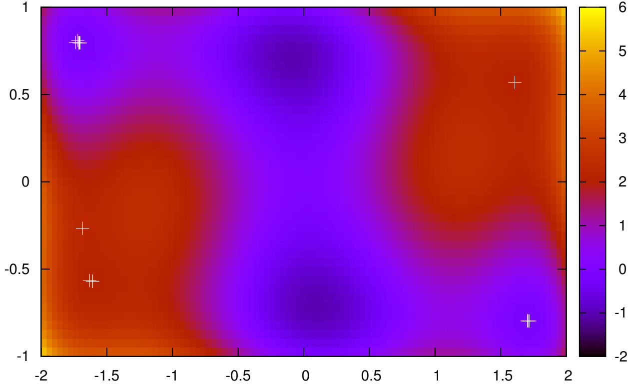
$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$



Multistart with IPOPT, $k = 10$

Six-hump camelback function

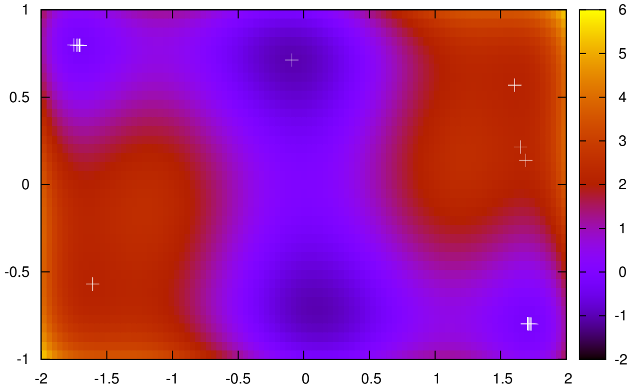
$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$



Multistart with IPOPT, $k = 20$

Six-hump camelback function

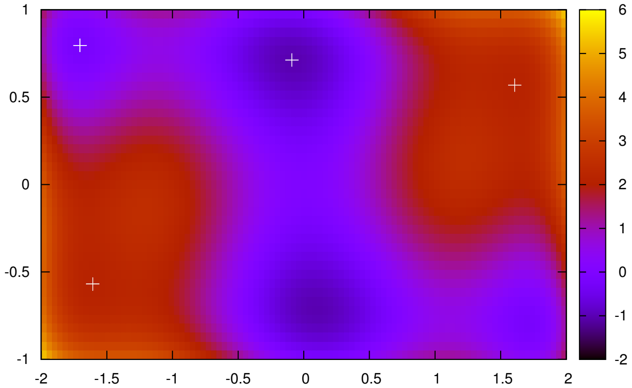
$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$



Multistart with IPOPT, $k = 50$

Six-hump camelback function

$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$$



Multistart with SNOPT, $k = 20$

Outline

1 Motivating Applications

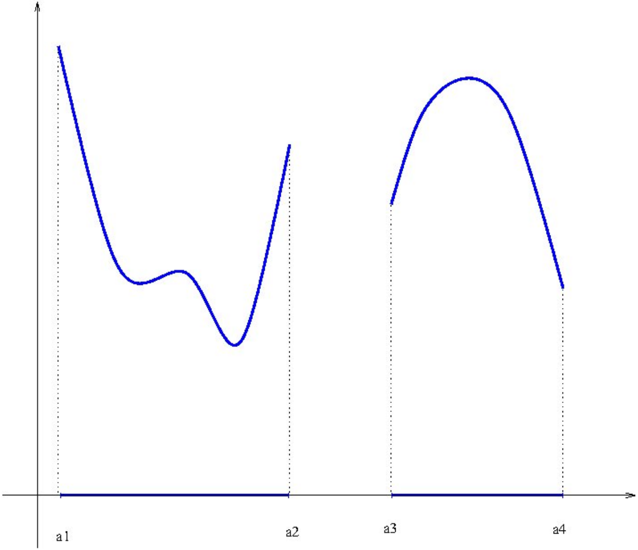
2 Global Optimization methods
- Multistart
- Spatial Branch-and-Bound
  - Standard form
  - Convexification
  - Expression trees
  - Variable ranges
  - Bounds tightening
  - Reformulation Linearization Technique (RLT)

Spatial Branch-and-Bound

Falk and Soland (1969) "An algorithm for separable nonconvex programming problems".
20 years ago: first general-purpose "exact" algorithms for nonconvex MINLP.

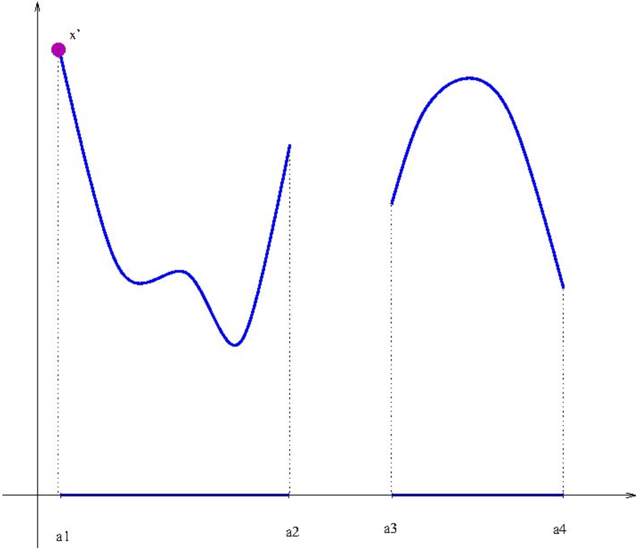
- **Tree-like search**
- Explores search space exhaustively but **implicitly**
- Builds a sequence of **decreasing upper bounds** and **increasing lower bounds** to the global optimum
- Exponential worst-case
- Only general-purpose **"exact"** algorithm for MINLP
  Since continuous vars are involved, should say **"ε-approximate"**
- Like BB for MILP, but may branch on continuous vars
  Done whenever one is involved in a nonconvex term

Spatial B&B: Example



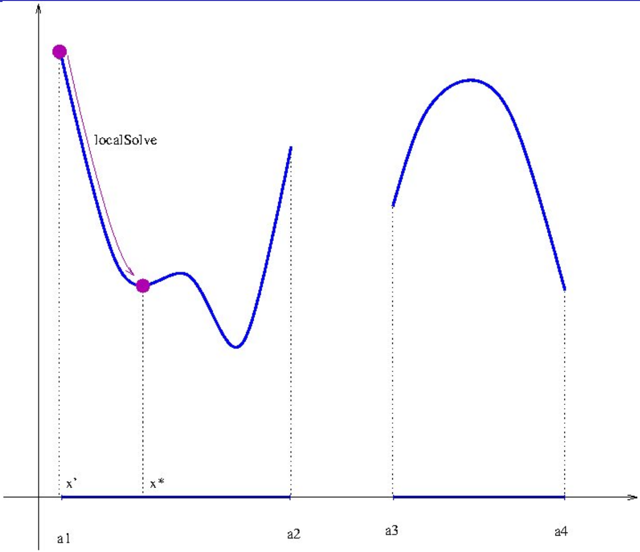
Original problem P

Spatial B&B: Example



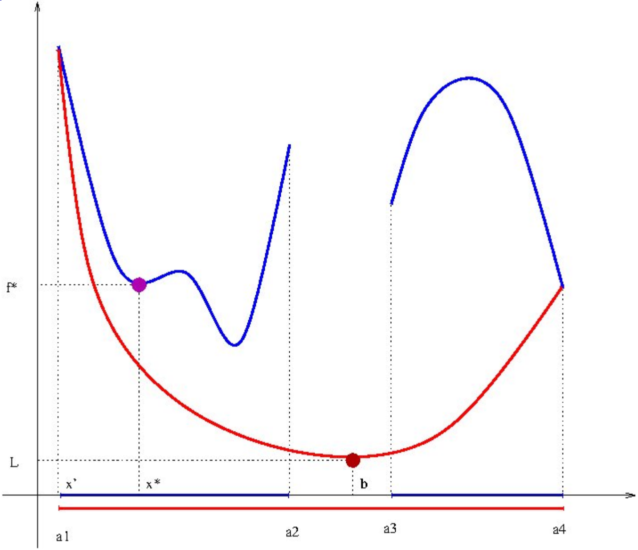
Starting point x'

Spatial B&B: Example



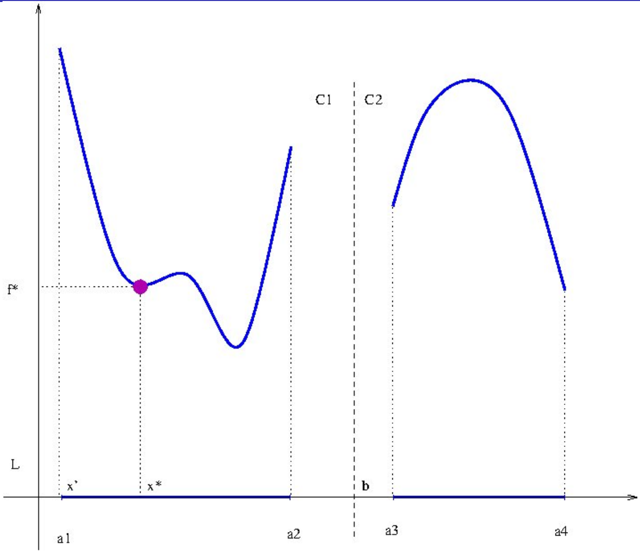
Local (upper bounding) solution x^*

Spatial B&B: Example



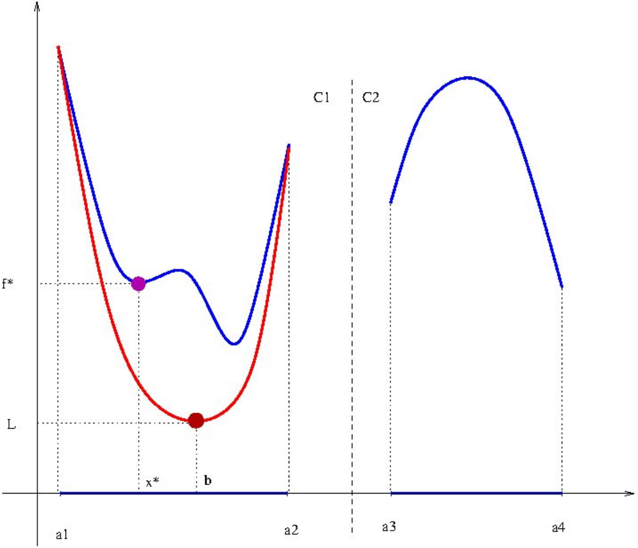
Convex relaxation (lower) bound $\bar{f}$ with $|f^* - \bar{f}| > \varepsilon$

Spatial B&B: Example



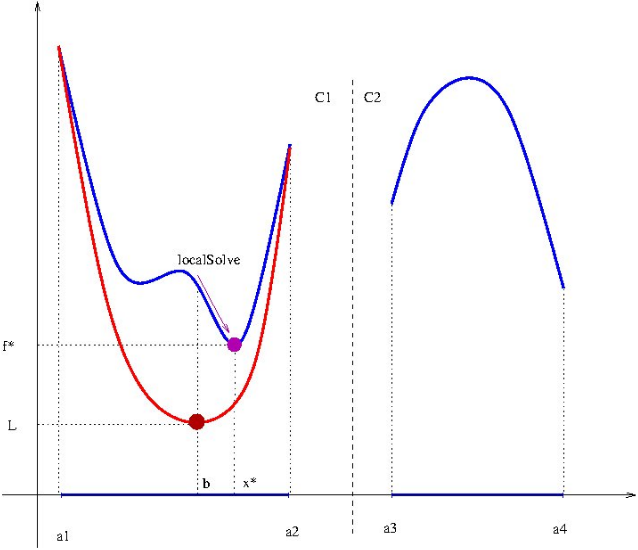
Branch at $x = \bar{x}$ into C_1, C_2

Spatial B&B: Example



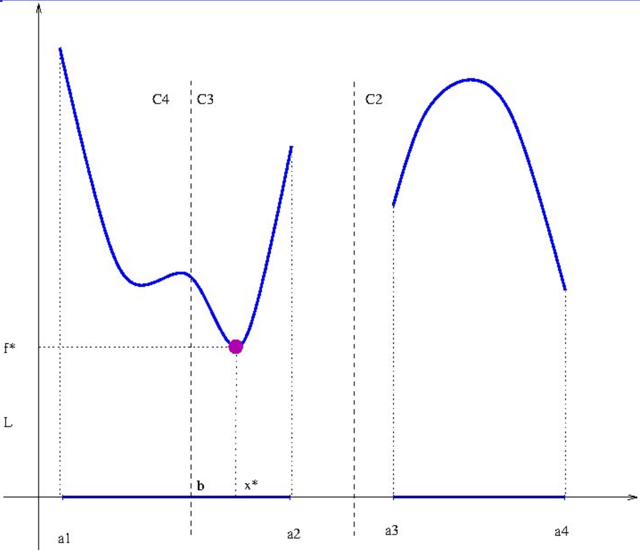
Convex relaxation on C_1: lower bounding solution $\bar{x}$

Spatial B&B: Example



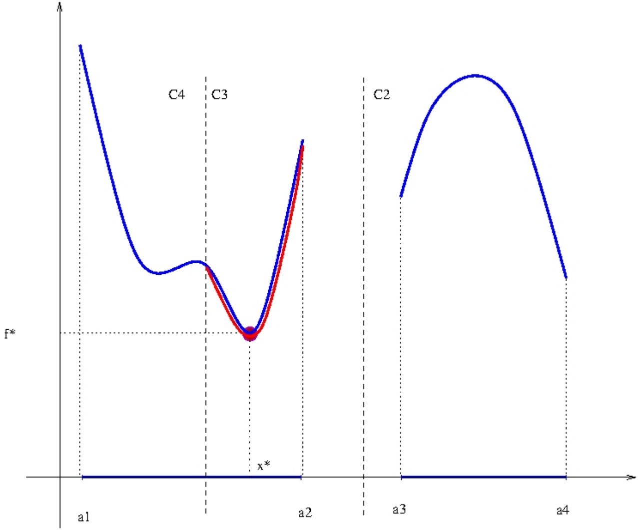
localSolve. *from $\bar{x}$: new upper bounding solution x^**

Spatial B&B: Example



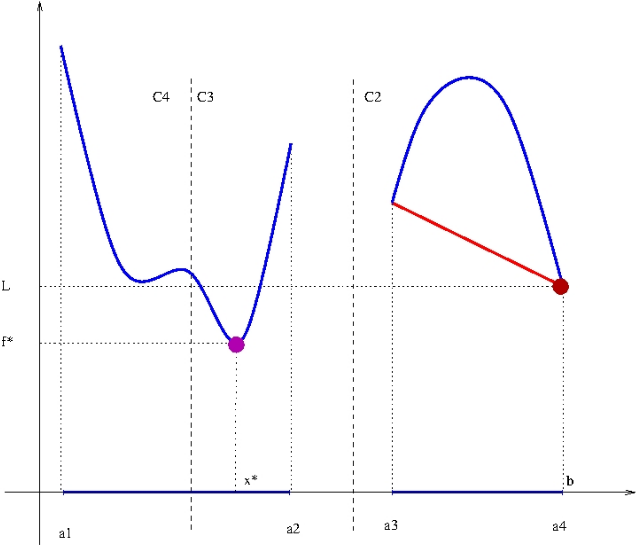
$|f^* - \bar{f}| > \varepsilon$: *branch at* $x = \bar{x}$

Spatial B&B: Example

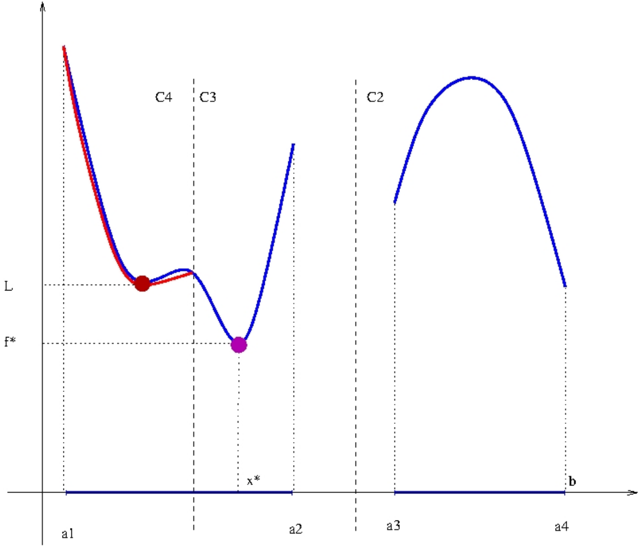


Repeat on C_3: get $\bar{x} = x^*$ and $|f^* - \bar{f}| < \varepsilon$, no more branching

Spatial B&B: Example

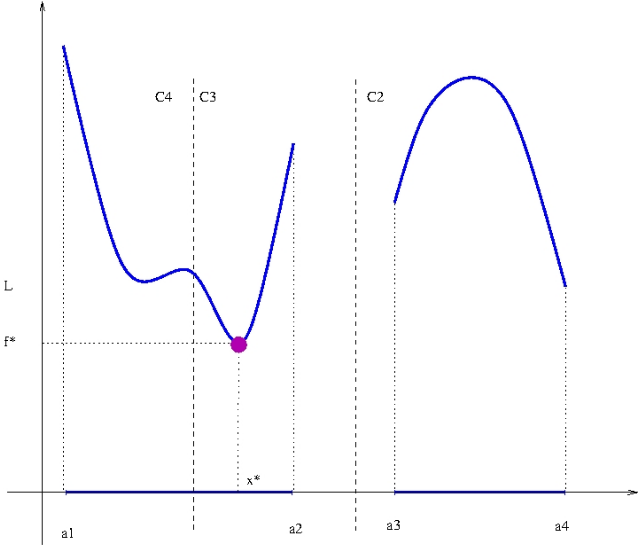


Repeat on C_2: $\bar{f} > f^*$ (can't improve x^* in C_2)



Repeat on C_4: $\bar{f} > f^*$ (can't improve x^* in C_4)

Spatial B&B: Example



No more subproblems left, return x^* and terminate

Spatial B&B: Pruning

1. P was branched into C_1, C_2
2. C_1 was branched into C_3, C_4
3. C_3 was **pruned by optimality**
   ($x^* \in \mathcal{G}(C_3)$ was found)
4. C_2, C_4 were **pruned by bound**
   (lower bound for C_2 worse than f^*)
5. No more nodes: whole space explored, $x^* \in \mathcal{G}(P)$

- Search generates a tree
- Suproblems are nodes
- Nodes can be pruned by optimality, bound or **infeasibility** (when subproblem is infeasible)
- Otherwise, they are branched

Spatial B&B: General idea

Aimed at solving “factorable functions”, i.e., f and g of the form:

$$\sum_h \prod_k f_{hk}(x, y)$$

where $f_{hk}(x, y)$ are univariate functions $\forall h, k$.

Spatial B&B: General idea

Aimed at solving "factorable functions", i.e., f and g of the form:

$$\sum_h \prod_k f_{hk}(x, y)$$

where $f_{hk}(x, y)$ are univariate functions $\forall h, k$.

- Exact reformulation of MINLP so as to have **"isolated basic nonlinear functions"** (additional variables and constraints).

Spatial B&B: General idea

Aimed at solving "factorable functions", i.e., f and g of the form:

$$\sum_h \prod_k f_{hk}(x, y)$$

where $f_{hk}(x, y)$ are univariate functions $\forall h, k$.

- Exact reformulation of MINLP so as to have **"isolated basic nonlinear functions"** (additional variables and constraints).
- Relax (linear/convex) the basic nonlinear terms (**library of envelopes/underestimators**).

Spatial B&B: General idea

Aimed at solving "factorable functions", i.e., f and g of the form:

$$\sum_h \prod_k f_{hk}(x, y)$$

where $f_{hk}(x, y)$ are univariate functions $\forall h, k$.

- Exact reformulation of MINLP so as to have **"isolated basic nonlinear functions"** (additional variables and constraints).
- Relax (linear/convex) the basic nonlinear terms (**library of envelopes/underestimators**).
- Relaxation depends on variable bounds, thus **branching** potentially strengthen it.

Outline

Spatial B&B: exact reformulation to standard form

Consider a NLP for simplicity. Transform it in a **standard form** like:

$$\begin{array}{rcll} \min c^\mathsf{T}(x, w) & & & \\ A(x, w) & \leq & b & \\ w_{ij} & = & x_i \bigotimes x_j & \text{for suitable } i, j \\ x & \in & X & \\ w & \in & W & \end{array}$$

where, for example, $\bigotimes \in$ {**sum, product, quotient, power, exp, log, sin, cos, abs**} (Couenne).

Outline

1 Motivating Applications

2 Global Optimization methods
- Multistart
- Spatial Branch-and-Bound
  - Standard form
  - Convexification
  - Expression trees
  - Variable ranges
  - Bounds tightening
  - Reformulation Linearization Technique (RLT)

Spatial B&B: convexification

Relax $w_{ij} = x_i \bigotimes x_j$ $\forall$ suitable i, j where $\bigotimes \in$ {sum, product, quotient, power, exp, log, sin, cos, abs} such that:

$$\begin{aligned} w_{ij} &\leq \text{overestimator}(x_i \bigotimes x_j) \\ w_{ij} &\geq \text{underestimator}(x_i \bigotimes x_j) \end{aligned}$$

Convex relaxation is **not the tightest possible**, but **built automatically** .

Spatial B&B: convexification

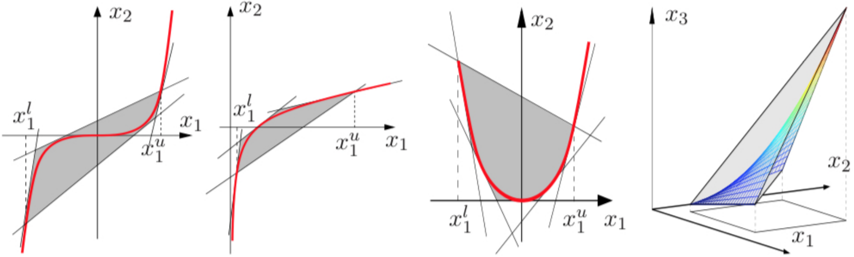
Relax $w_{ij} = x_i \bigotimes x_j$ $\forall$ suitable i, j where $\bigotimes \in$ {sum, product, quotient, power, exp, log, sin, cos, abs} such that:

$$
\begin{aligned}
w_{ij} &\leq \text{overestimator}(x_i \bigotimes x_j) \\
w_{ij} &\geq \text{underestimator}(x_i \bigotimes x_j)
\end{aligned}
$$

Convex relaxation is **not the tightest possible**, but **built automatically** .

- Underestimator/overestimator of convex/concave function: tangent cuts (OA)
- Odd powers or Trigonometric functions: separate intervals in which function is convex or concave and do as for convex/concave functions
- Product or Quotient: Mc Cormick relaxation

Spatial B&B: Examples of Convexifications



(a) $x_2 = x_1^3$ (b) $x_2 = \log x_1$ (c) $x_2 = x_1^2$ (d) $x_3 = x_1 x_2$

P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, “Branching and bounds tightening techniques for non-convex MINLP”. Optimization Methods and Software 24(4-5): 597-634 (2009).

Example: Standard Form Reformulation

$$\begin{aligned} \min x_1^2 + x_1 x_2 & & \\ x_1 + x_2 &\geq& 1 \\ x_1 &\in& [0, 1] \\ x_2 &\in& [0, 1] \end{aligned}$$

Example: Standard Form Reformulation

$$\begin{aligned} \min x_1^2 + x_1 x_2 & \\ x_1 + x_2 &\geq 1 \\ x_1 &\in [0,1] \\ x_2 &\in [0,1] \end{aligned}$$

becomes

$$\begin{aligned} \min w_1 + w_2 & \\ w_1 &= x_1^2 \\ w_2 &= x_1 x_2 \\ x_1 + x_2 &\geq 1 \\ x_1 &\in [0,1] \\ x_2 &\in [0,1] \end{aligned}$$

Example: .mod from Couenne

```
var x1 <= 1, >= 0;
var x2 <= 1, >= 0;

minimize of:
x1**2 + x1*x2;
subject to constraint:
x1 + x2 >= 1;
```

Example: .mod from Couenne

var x1 <= 1, >= 0;
var x2 <= 1, >= 0;

minimize of:
x1**2 + x1*x2;
subject to constraint:
x1 + x2 >= 1;

```
# Problem name: extended-aw.mod

# original variables

var x_0 >= 0 <= 1 default 0;
var w_1 >= 0 <= 1 default 1;
var w_2 >= 0 <= 1 default 0;
var w_3 >= 0 <= 1 default 0;
var w_4 >= 0 <= 2 default 0;

# objective

minimize obj: w_4;

# aux. variables defined

aux1: w_1 = (1-x_0);
aux2: w_2 = (x_0**2);
aux3: w_3 = (x_0*w_1);
aux4: w_4 = (w_2+w_3);

# constraints
```

Convex hull of pieces weaker than the whole convex hull

Consider the following feasible set:

$$\begin{aligned} x_1^2 + x_2^2 &\geq 1 \\ x_1, x_2 &\in [0, 2] \end{aligned}$$

Convex hull of pieces weaker than the whole convex hull

Consider the following feasible set:

$$\begin{aligned} x_1^2 + x_2^2 &\geq 1 \\ x_1, x_2 &\in [0, 2] \end{aligned}$$

Convex hull: $x_1 + x_2 \geq 1$

Convex hull of pieces weaker than the whole convex hull

Consider the following feasible set:

$$\begin{aligned} x_1^2 + x_2^2 &\geq 1 \\ x_1, x_2 &\in [0,2] \end{aligned}$$

Convex hull: $x_1 + x_2 \geq 1$

Convex hull of standard form

$$\begin{aligned} x_3 + x_4 &\geq 1 \\ x_3 &\leq x_1^2 \\ x_4 &\leq x_1^2 \\ x_1, x_2 &\in [0,2] \end{aligned}$$

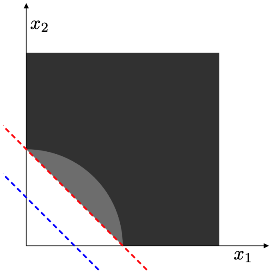


Figure: Source Belotti et al. (2013)

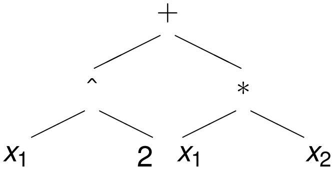
Outline

Expression trees

Representation of objective f and constraints g

Encode mathematical expressions in trees or DAGs

E.g. $x_1^2 + x_1 x_2$:

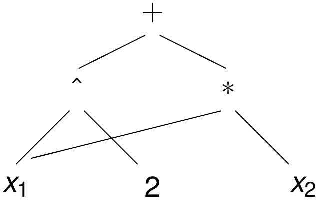


Expression trees

Representation of objective f and constraints g

Encode mathematical expressions in trees or DAGs

E.g. $x_1^2 + x_1 x_2$:



Outline

Variable ranges

- Crucial property for sBB convergence: **convex relaxation tightens as variable range widths decrease**
- convex/concave under/over-estimator constraints are (convex) functions of x^L, x^U
- it makes sense to **tighten** x^L, x^U at the sBB root node (trading off speed for efficiency) and at each other node (trading off efficiency for speed)

Outline

Bounds Tightening

- In sBB we need to tighten variable bounds at each node
- Two methods:
  - Optimization Based Bounds Tightening (OBBT)
  - Feasibility Based Bounds Tightening (FBBT)
- OBBT:
  for each variable x in P compute
  - $\underline{x} = \min\{x \mid \text{conv. rel. constr.}\}$
  - $\overline{x} = \max\{x \mid \text{conv. rel. constr.}\}$

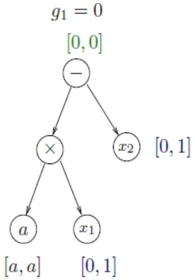
  Set $\underline{x} \leq x \leq \overline{x}$

Bounds Tightening

- In sBB we need to tighten variable bounds at each node
- Two methods:
  - Optimization Based Bounds Tightening (OBBT)
  - Feasibility Based Bounds Tightening (FBBT)
- FBBT:

propagation of intervals up and down constraint expression trees, with tightening at the root node

Example: $5x_1 - x_2 = 0$.



Bounds Tightening

- In sBB we need to tighten variable bounds at each node
- Two methods:
  - Optimization Based Bounds Tightening (OBBT)
  - Feasibility Based Bounds Tightening (FBBT)
- FBBT: **propagation of intervals up and down constraint expression trees, with tightening at the root node**

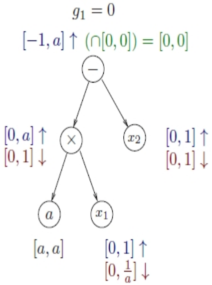
  Example: $5x_1 - x_2 = 0$.

  Up: $\otimes$:$[5,5] \times [0,1] = [0,5]$; $\ominus$:$[0,5] - [0,1] = [-1,5]$.

  Root node tightening: $[-1,5] \cap [0,0] = [0,0]$.

  Downwards: $\otimes$:$[0,0] + [0,1] = [0,1]$;

  x_1:$[0,1]/[5,5] = [0, \frac{1}{5}]$



Outline

1. Motivating Applications

2. Global Optimization methods
   - Multistart
   - Spatial Branch-and-Bound
     - Standard form
     - Convexification
     - Expression trees
     - Variable ranges
     - Bounds tightening
     - Reformulation Linearization Technique (RLT)

RLT: Quadratic problems

- All nonlinear terms are quadratic monomials
- Aim to **reduce gap betwen the problem and its convex relaxation**
- $\Rightarrow$ replace quadratic terms with suitable linear constraints (fewer nonlinear terms to relax)
- Can be obtained by considering linear relations (called **reduced RLT constraints**) between original and linearizing variables

RLT: Quadratic problems

Hp. We have $w_{ij} = x_i x_j$ for all $i = j = 1, \ldots, n$.
How to **strengthen the relaxation** obtained by replacing $w_{ij} = x_i x_j$ with its McCormick envelops?

RLT: Quadratic problems

Hp. We have $w_{ij} = x_i x_j$ for all $i = j = 1, \ldots, n$.
How to **strengthen the relaxation** obtained by replacing $w_{ij} = x_i x_j$ with its McCormick envelops?

- Take a variable x_k and a constraint $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$

RLT: Quadratic problems

Hp. We have $w_{ij} = x_i x_j$ for all $i = j = 1, \ldots, n$.
How to **strengthen the relaxation** obtained by replacing $w_{ij} = x_i x_j$ with its McCormick envelops?

- Take a variable x_k and a constraint $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$
- Multiply both the LHS and RHS of the constraint by x_i:

$$\sum_{j=1}^{n} a_{ij} x_j x_k \leq b_i x_k$$

RLT: Quadratic problems

Hp. We have $w_{ij} = x_i x_j$ for all $i = j = 1, \ldots, n$.
How to **strengthen the relaxation** obtained by replacing $w_{ij} = x_i x_j$ with its McCormick envelops?

- Take a variable x_k and a constraint $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$
- Multiply both the LHS and RHS of the constraint by x_i:

$$\sum_{j=1}^{n} a_{ij} x_j x_k \leq b_i x_k$$

- Linearize the resulting constraint by replacing $x_j x_k$ with w_{jk}:

$$\sum_{j=1}^{n} a_{ij} w_{jk} \leq b_i x_k$$

Example: pooling problem

Q-formulation

$$\sum_{j \in J_i} x_{ij} + \sum_{l \in L_i} x_{il} \le C_i, \qquad \forall i \in I$$

$$\sum_{j \in J_l} x_{lj} \le C_l, \qquad \forall l \in L$$

$$\sum_{i \in I_j} x_{ij} + \sum_{l \in L_j} x_{lj} \le C_j, \qquad \forall j \in J$$

$$x_{il} - q_{il} \sum_{j \in J_l} x_{lj} = 0 \qquad \forall i \in I, l \in L_i$$

$$\sum_{i \in I_l} q_{il} = 1 \qquad \forall l \in L$$

$$\sum_{i \in I_j} \lambda_{ki} x_{ij} + \sum_{l \in L_j} x_{lj} \left(\sum_{i \in I_l} \lambda_{ki} q_{il} \right) \le \alpha_{kj} \sum_{i \in I_j \cup L_j} x_{ij}, \quad \forall k \in K, j \in J$$

Example: pooling problem

PQ-formulation by Sahinidis and Tawarmalani (2005).
Like Q-formulation but with extra (redundant) constraints:

- $x_{lj} \sum_{i \in I_l} q_{il} = x_{lj} \quad \forall l \in L, j \in J_l$
- $q_{il} \sum_{j \in J_l} x_{lj} \leq C_l q_{il} \quad \forall i \in I, l \in L_i$

Example: pooling problem

PQ-formulation by Sahinidis and Tawarmalani (2005).
Like Q-formulation but with extra (redundant) constraints:

- $x_{lj} \sum_{i \in I_l} q_{il} = x_{lj} \quad \forall l \in L, j \in J_l$
- $q_{il} \sum_{j \in J_l} x_{lj} \leq C_l q_{il} \quad \forall i \in I, l \in L_i$

One of the strongest known formulation!

Citations

- Sherali, Alameddine, *A new reformulation-linearization technique for bilinear programming problems*, JOGO, 1991
- Falk, Soland, *An algorithm for separable nonconvex programming problems*, Manag. Sci. 1969.
- Horst, Tuy, *Global Optimization*, Springer 1990.
- Ryoo, Sahinidis, *Global optimization of nonconvex NLPs and MINLPs with applications in process design*, Comp. Chem. Eng. 1995.
- Adjiman, Floudas et al., *A global optimization method, αBB, for general twice-differentiable nonconvex NLPs*, Comp. Chem. Eng. 1998.
- Smith, Pantelides, *A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs*, Comp. Chem. Eng. 1999.
- Nowak, *Relaxation and decomposition methods for Mixed Integer Nonlinear Programming*, Birkhäuser, 2005.
- Belotti, et al., *Branching and bounds tightening techniques for nonconvex MINLP*, Opt. Meth. Softw., 2009.
- Vigerske, PhD Thesis: Decomposition of Multistage Stochastic Programs and a Constraint Integer Programming Approach to Mixed-Integer Nonlinear Programming, Humboldt-University Berlin, 2013.