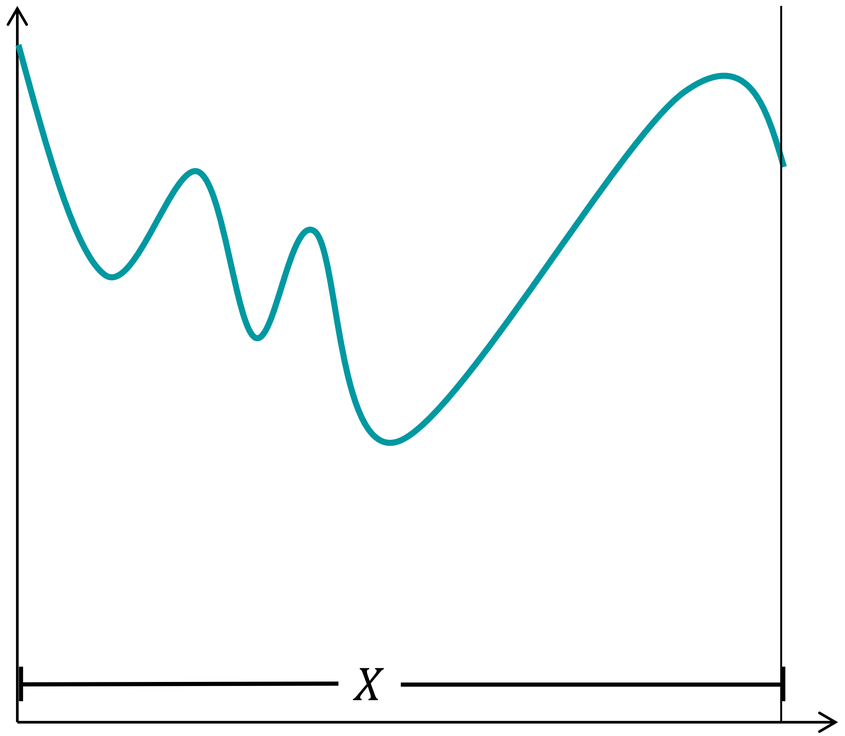
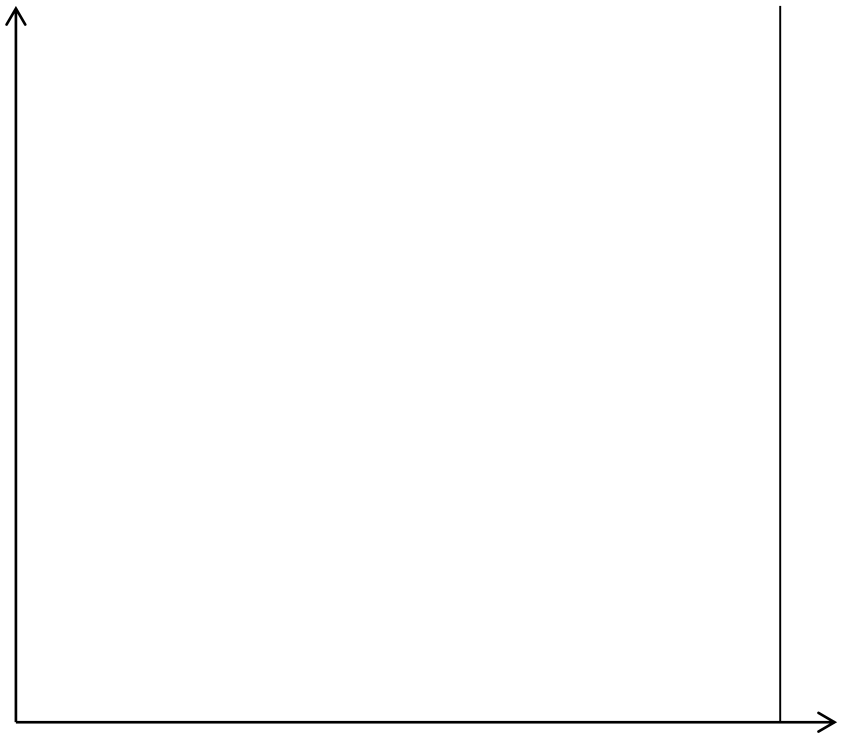
Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

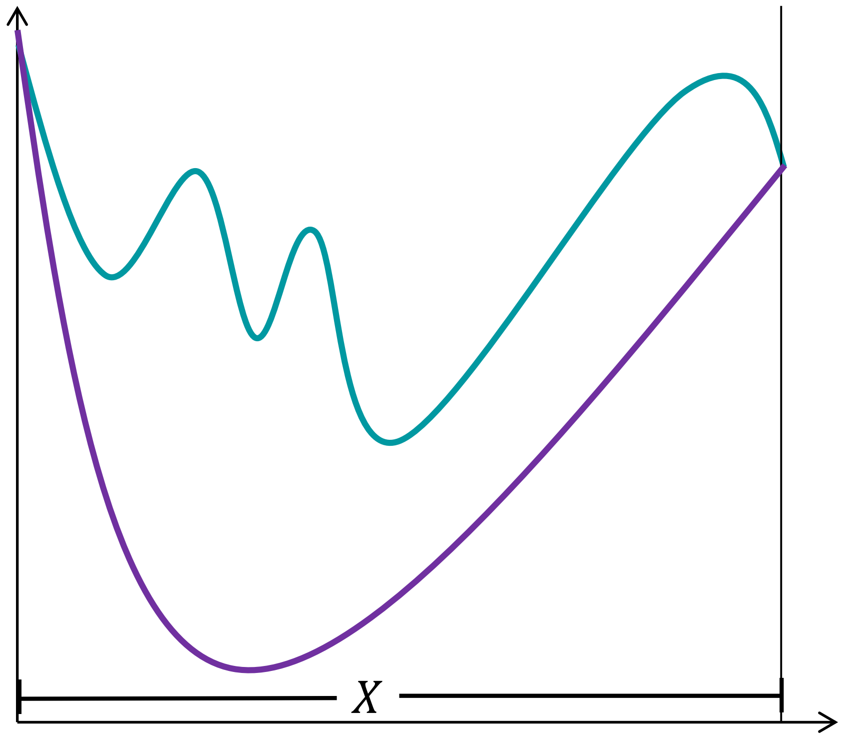
Branch & Bound for NLP

B&B Illustration for Box-Constrained NLPs (1)

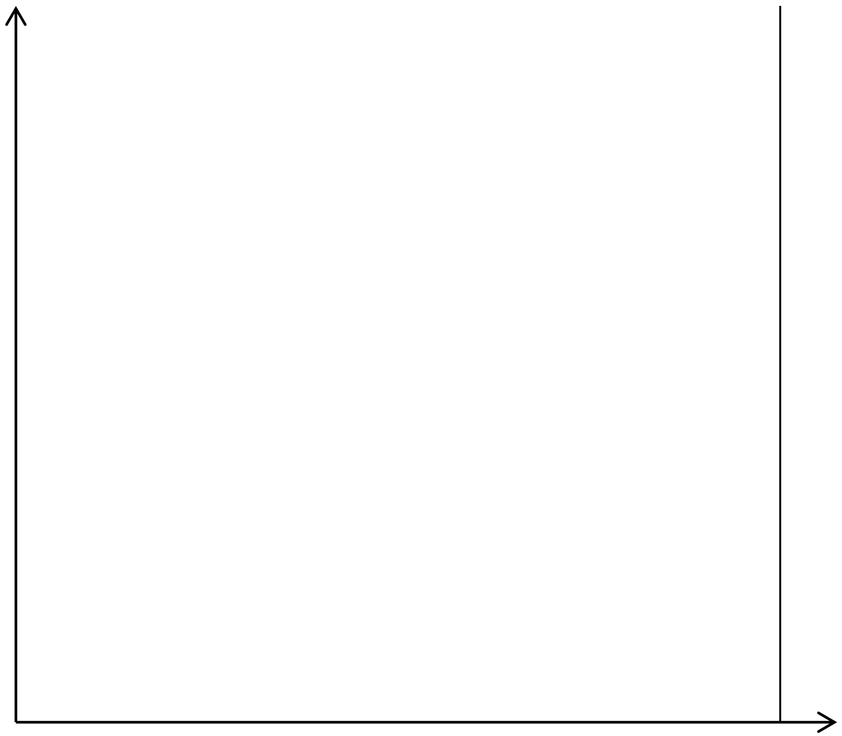




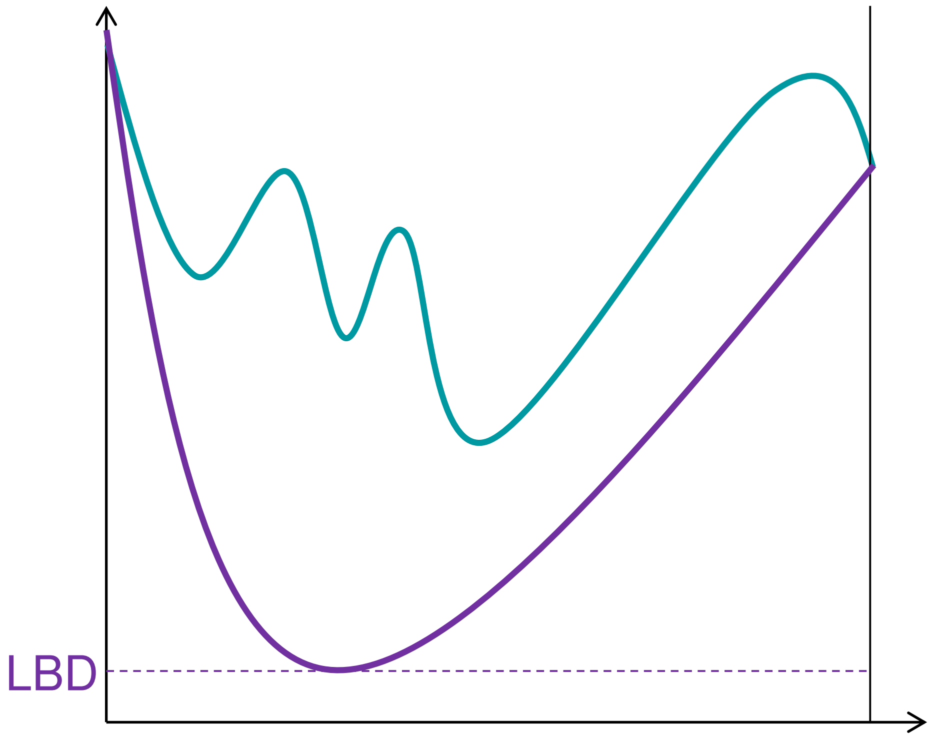
B&B Illustration for Box-Constrained NLPs (1)



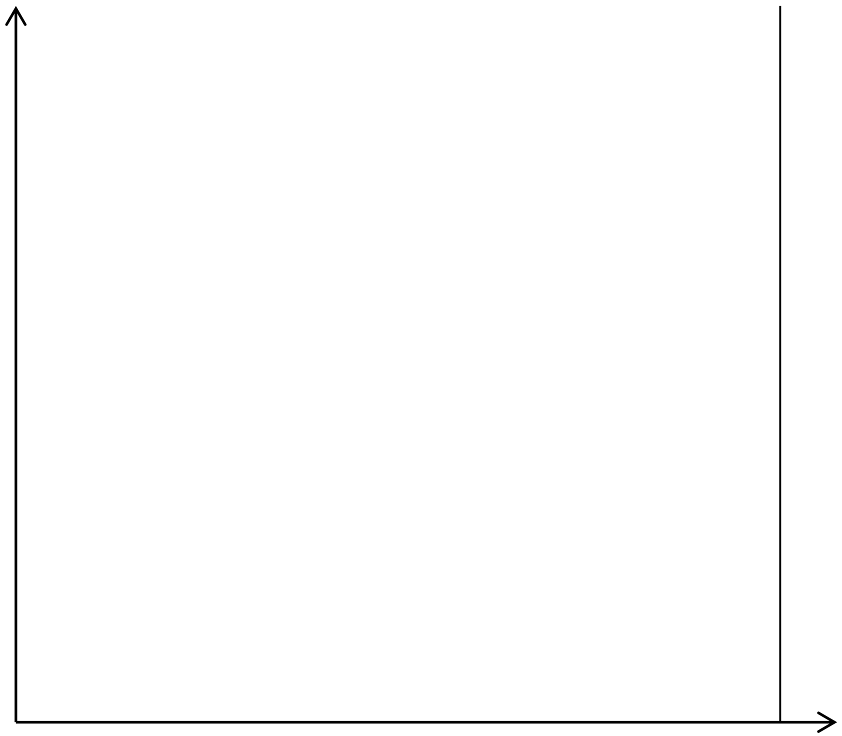
1. Construct a relaxation



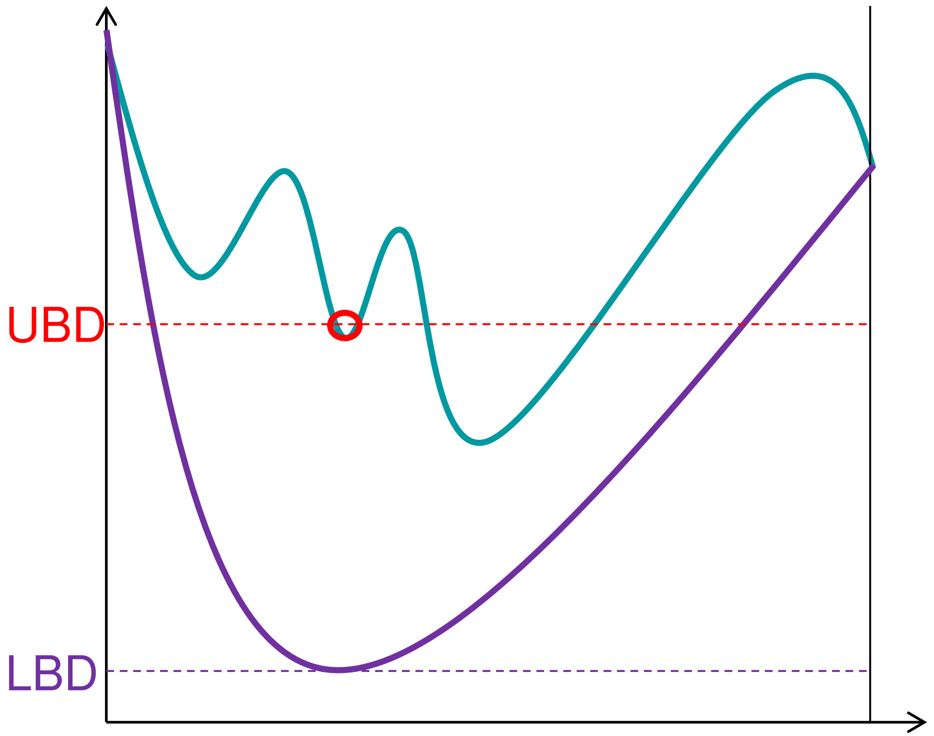
B&B Illustration for Box-Constrained NLPs (1)

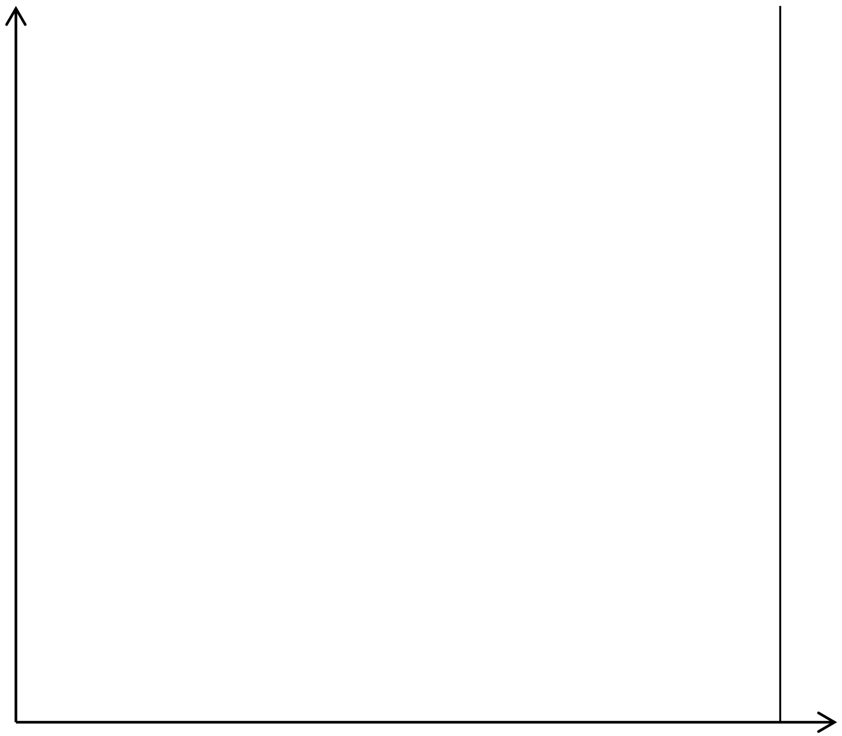


1. Construct a relaxation
2. Solve relaxation → LBD



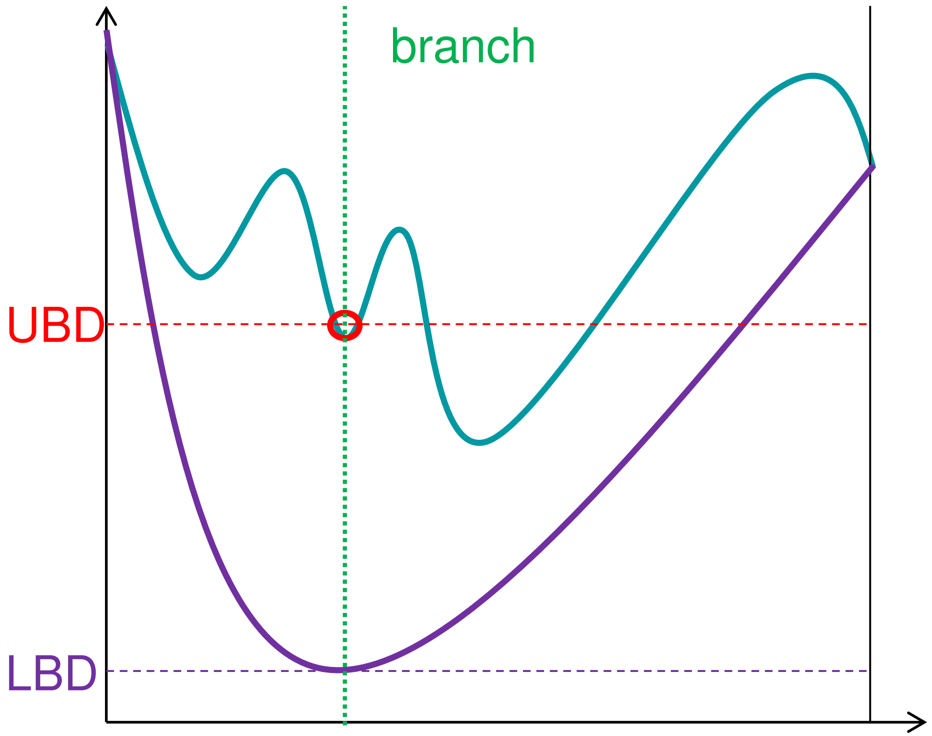
B&B Illustration for Box-Constrained NLPs (1)

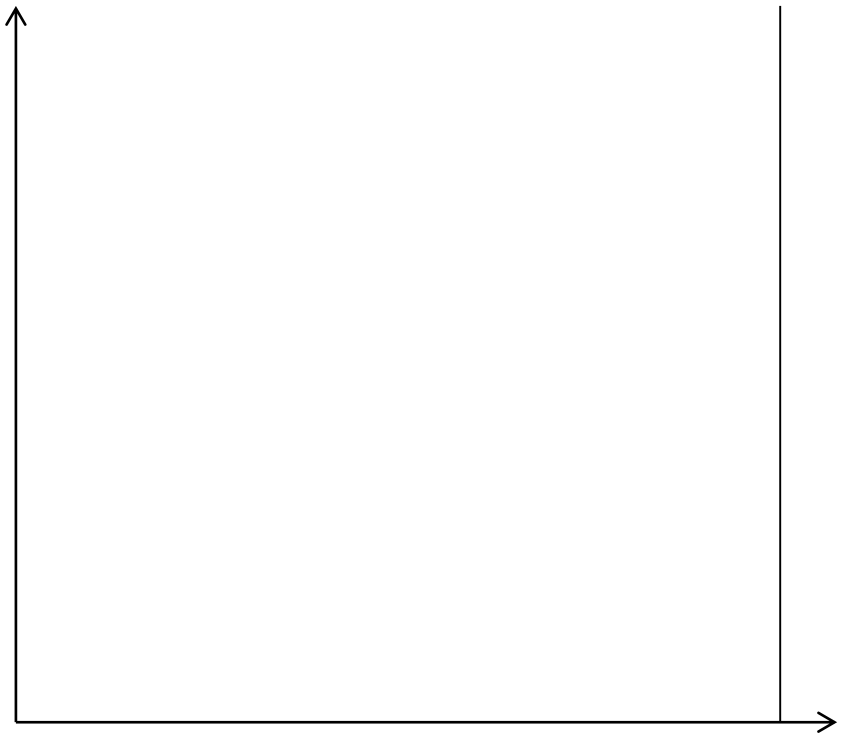




1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD

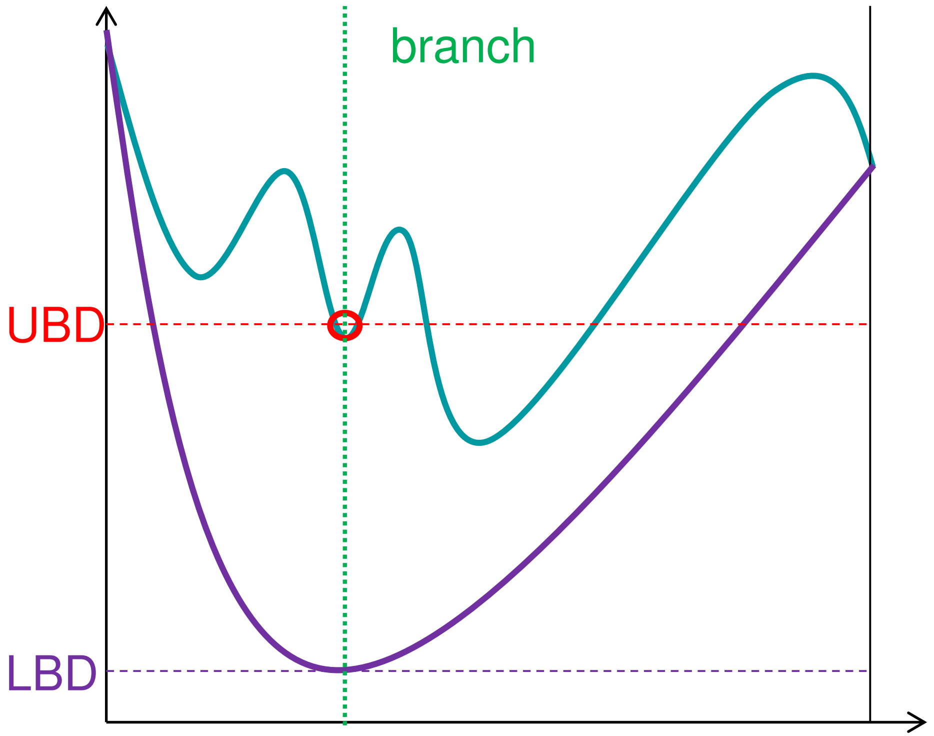
B&B Illustration for Box-Constrained NLPs (1)



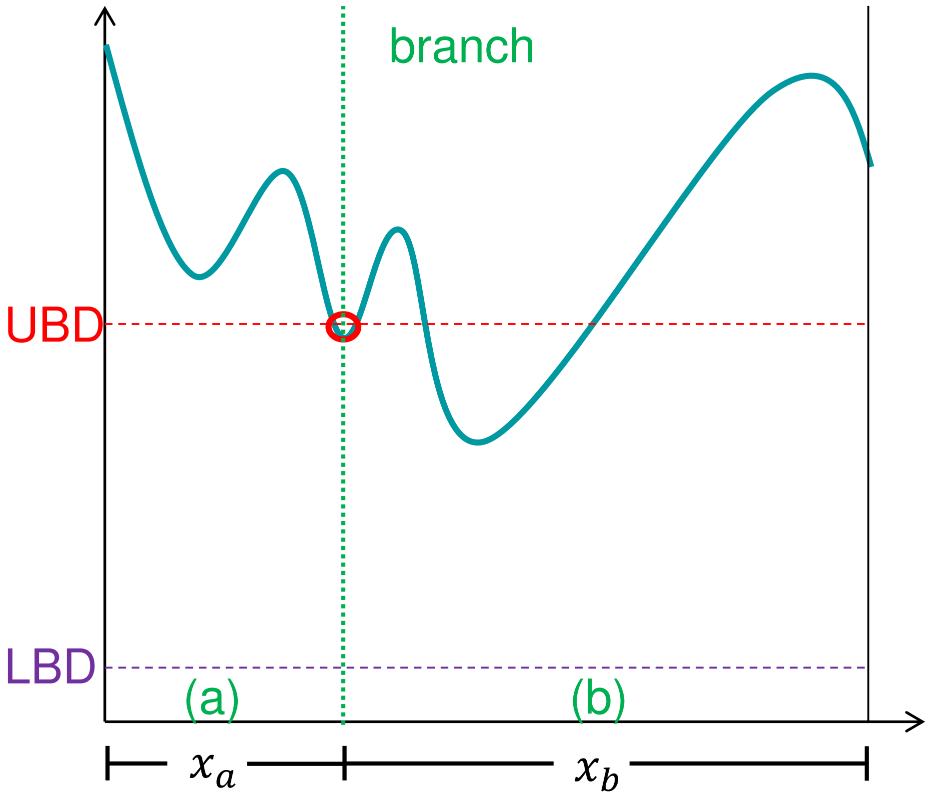


1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)

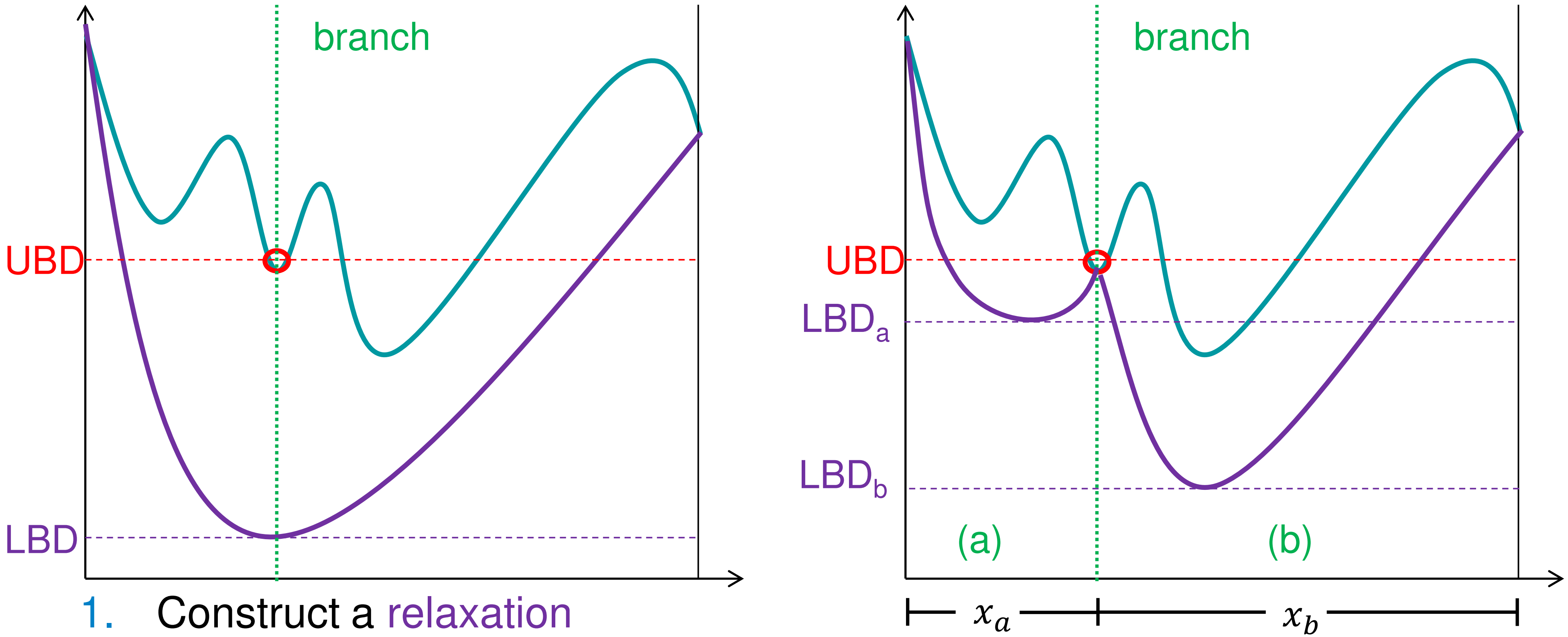
B&B Illustration for Box-Constrained NLPs (1)



1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)
5. Repeat steps for each node

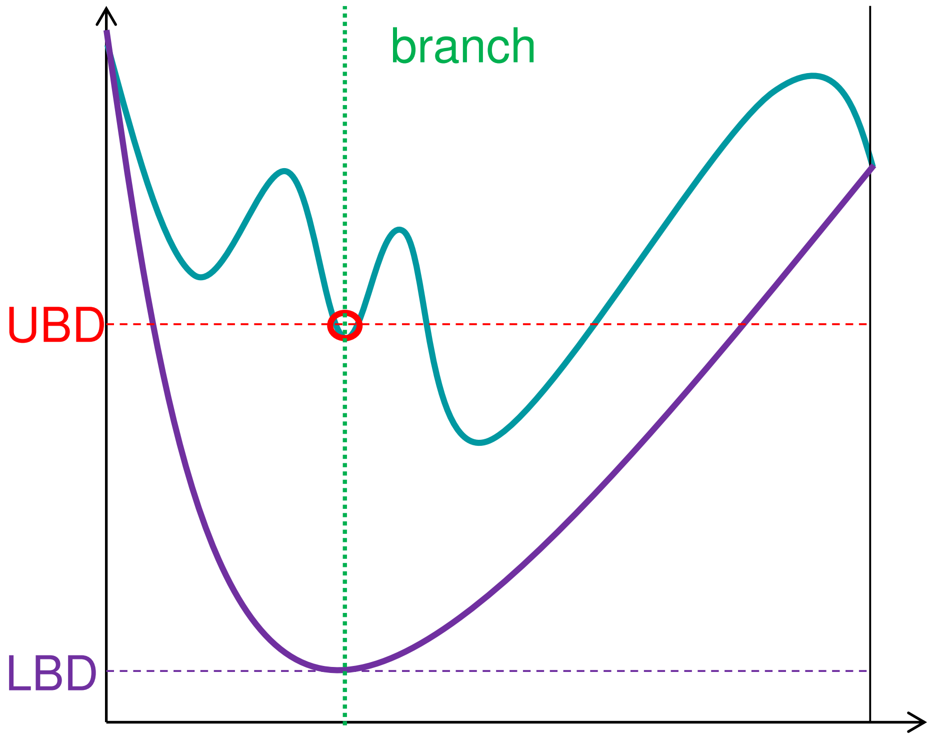


B&B Illustration for Box-Constrained NLPs (1)

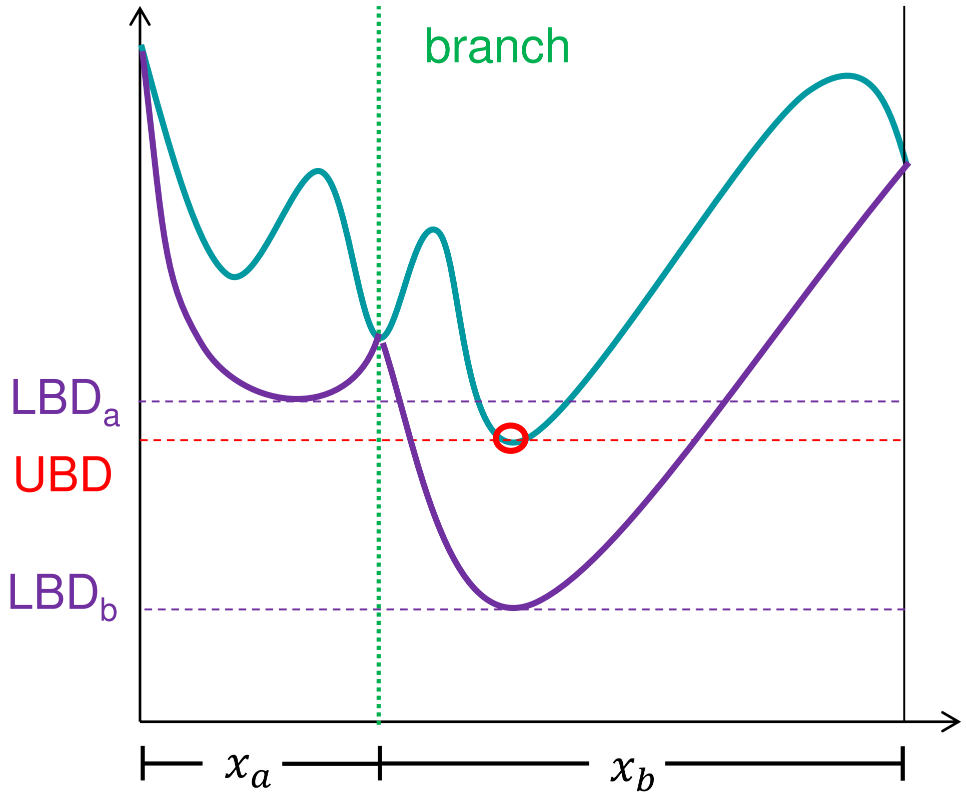


1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)
5. Repeat steps for each node

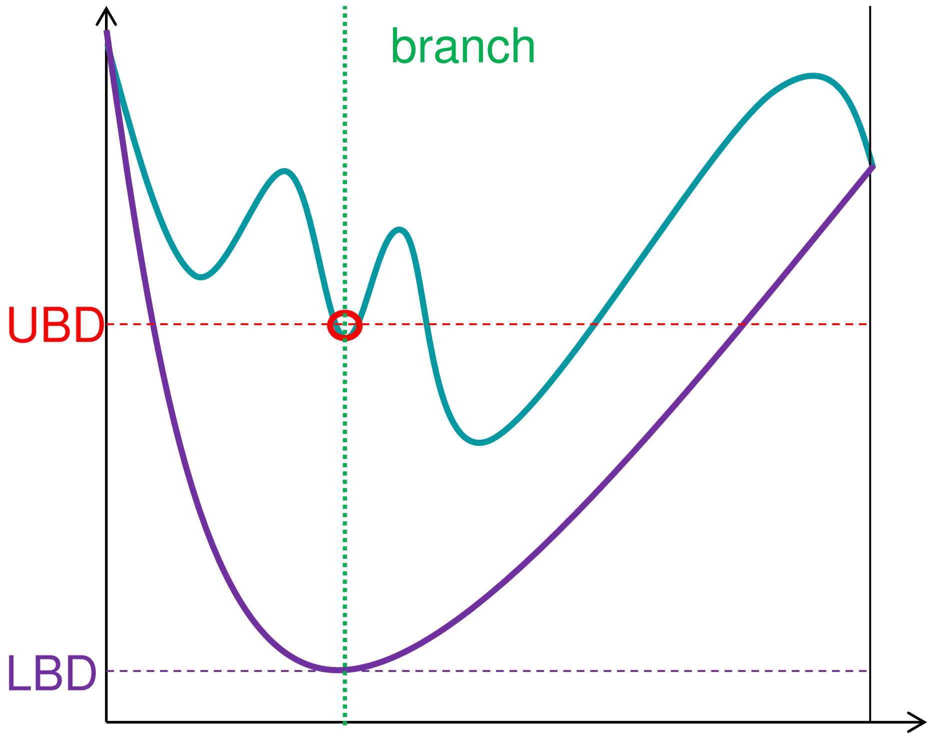
B&B Illustration for Box-Constrained NLPs (1)



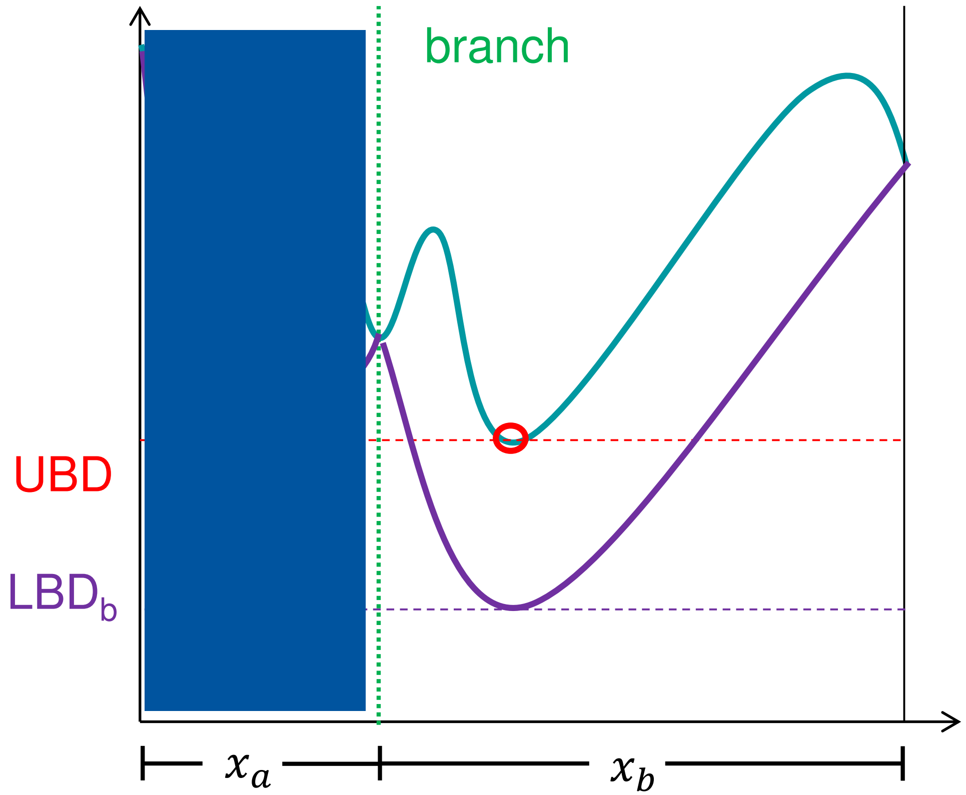
1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)
5. Repeat steps for each node



B&B Illustration for Box-Constrained NLPs (1)

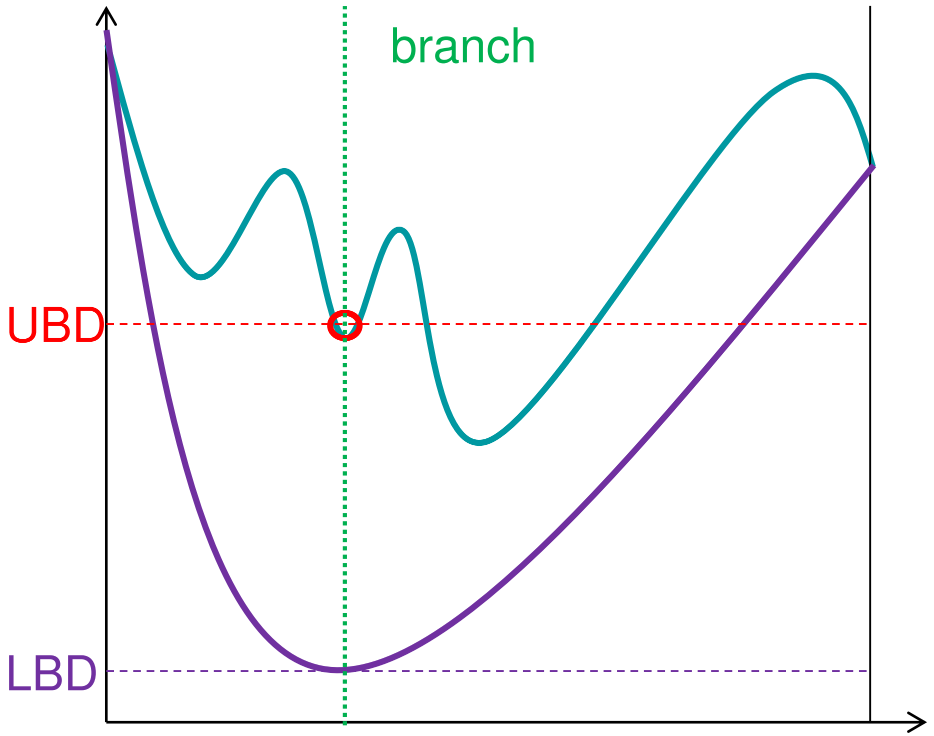


1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)
5. Repeat steps for each node

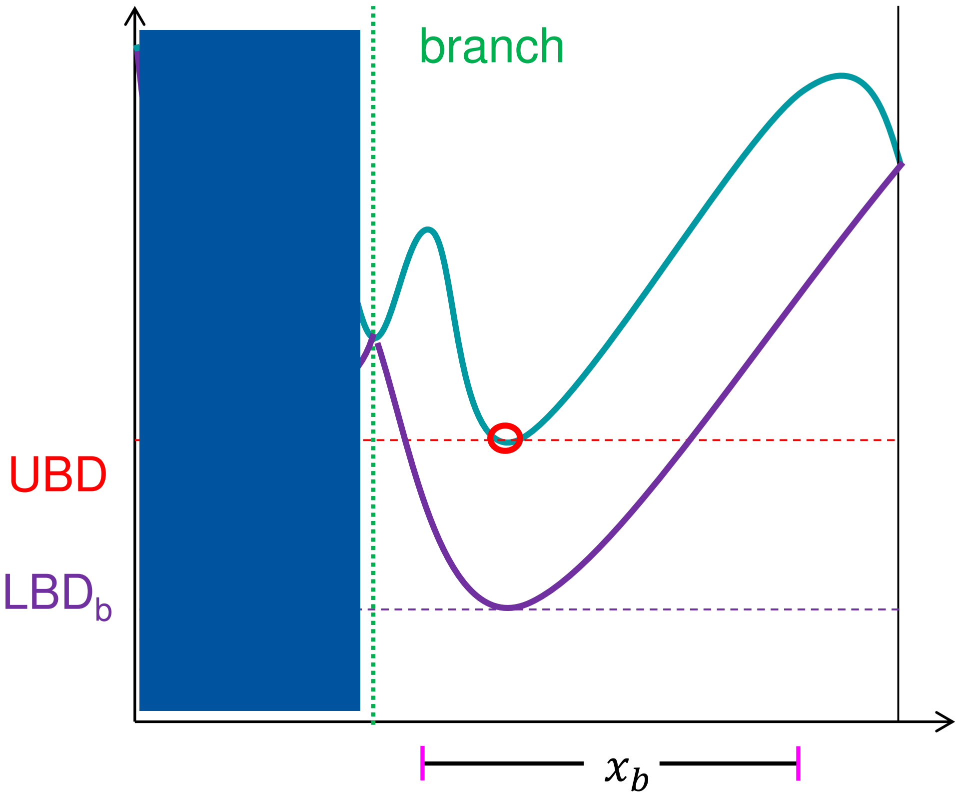


6. Fathom by value dominance

B&B Illustration for Box-Constrained NLPs (1)



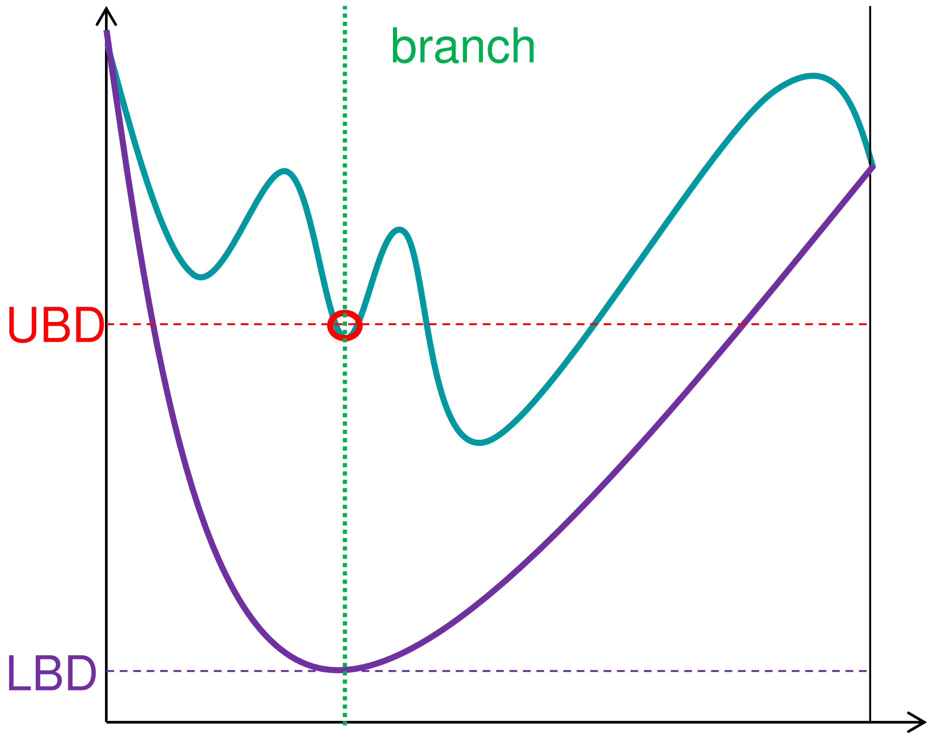
1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)
5. Repeat steps for each node



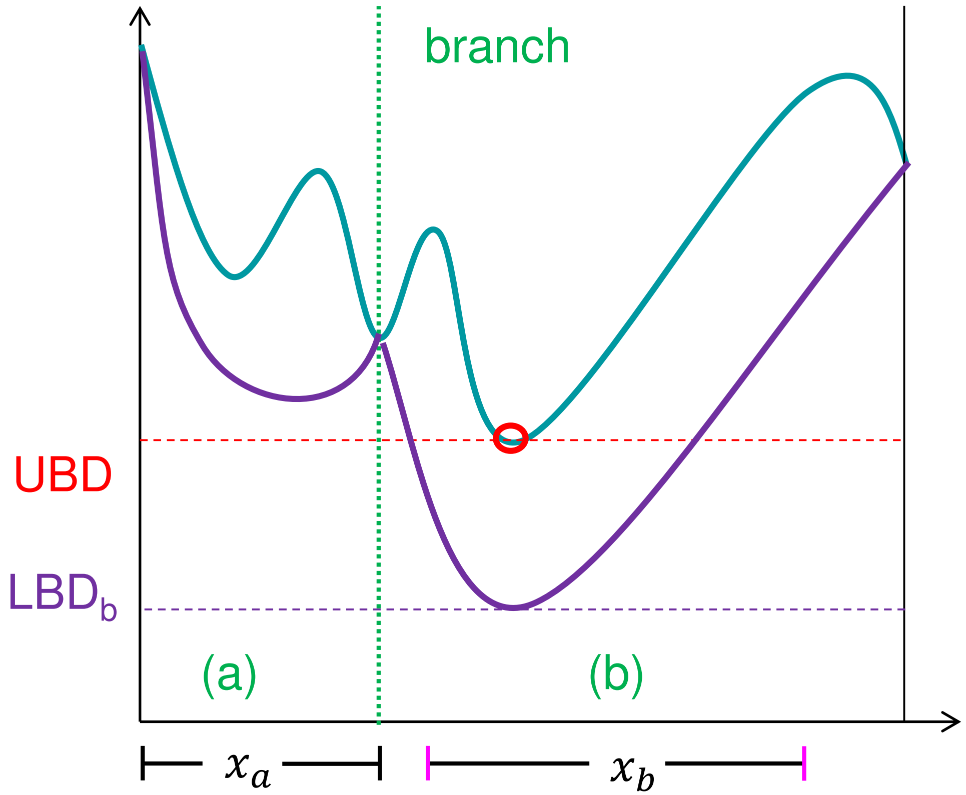
6. Fathom by value dominance

Range reduction of variables

B&B Illustration for Box-Constrained NLPs (1)



1. Construct a relaxation
2. Solve relaxation → LBD
3. Solve original locally → UBD
4. Branch to nodes (a) and (b)
5. Repeat steps 1-4



- How to get lower bounds?
- How to get upper bounds?
- (Range reduction of the variable bounds?)

Check Yourself

- What are the implications of nonconvex objective function?
- What are the implications of nonconvex feasible set?
- Describe B&B for nonconvex optimization.

Applied Numerical Optimization

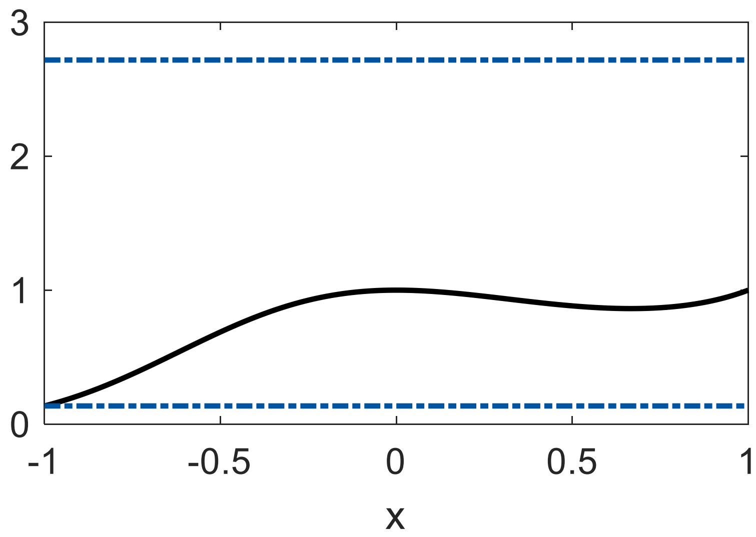
Prof. Alexander Mitsos, Ph.D.

Convex relaxations of nonconvex functions

Aachener Verfahrenstechnik
RWTH AACHEN UNIVERSITY

Basic Ideas for Relaxation of Functions: Natural Interval Extension

1. Decompose function to finite sequence of addition, multiplication and intrinsic functions
2. Propagate intervals of variables



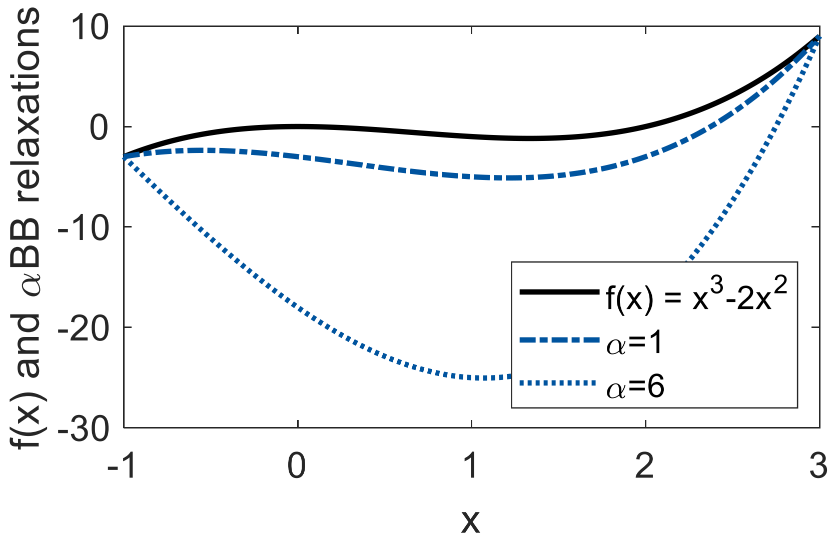
- Example: $\exp(x^3 - x^2)$ for $x \in [-1,1]$

$$\begin{aligned}
&\exp([-1,1]^3 - [-1,1]^2) \\
&\subset \exp([-1,1] - [0,1]) \\
&\subset \exp([-2,1]) \\
&\subset [\exp(-2)\,, \exp(1)]
\end{aligned}$$

- Applicable to most functions
- Simple and cheap but weak relaxations with linear convergence order
- Centered form and Taylor models are improvements

Basic Ideas for Relaxation of Functions: αBB Method

- αBB relaxations for smooth functions add a negative quadratic term
  $f(x) + \sum_i \alpha_i (x_i - x_i^L)(x_i - x_i^U)$
  - Relaxation for any $\alpha > 0$
  - Convex for large α



$x^L = -1$
$x^U = 3$

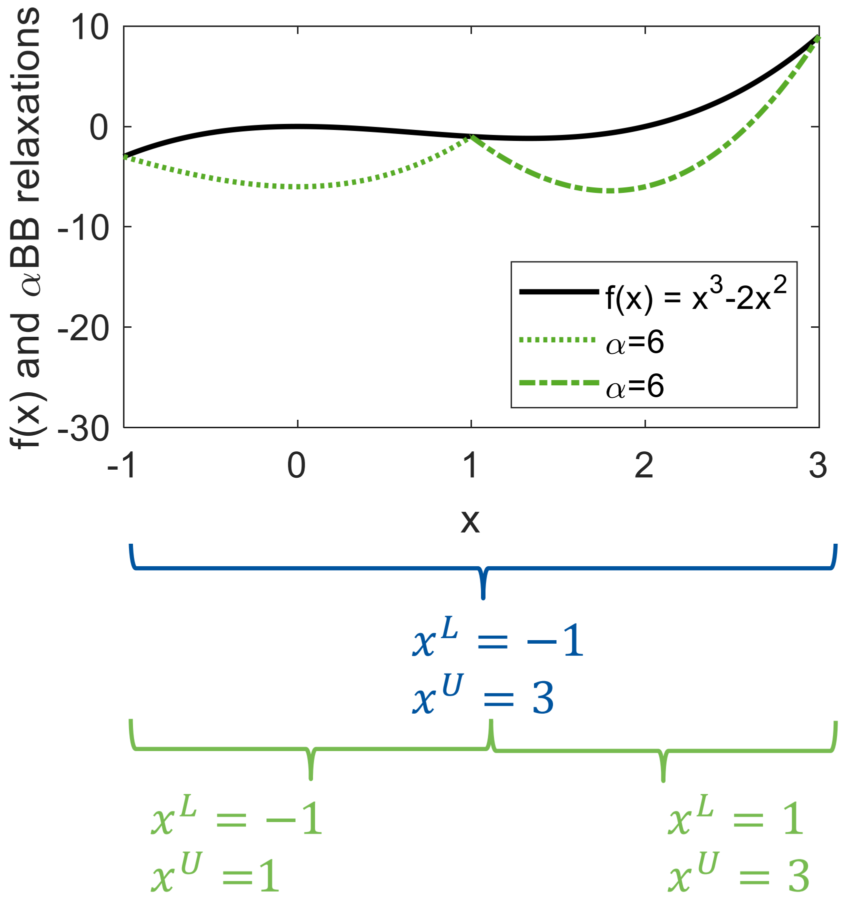
Maranas & Floudas JOGO 4.2 (1994): 135-170.
Akrotirianakis & Floudas JOGO 30.4 (2004): 367-390.

Basic Ideas for Relaxation of Functions: αBB Method

- αBB relaxations for smooth functions add a negative quadratic term
  $f(x) + \sum_i \alpha_i \, (x_i - x_i^L)(x_i - x_i^U)$
  - Relaxation for any $\alpha > 0$
  - Convex for large α
  - Calculate suitable α by underestimating eigenvalues of Hessian

- Quadratic convergence in $x_i^U - x_i^L$
  but often relatively weak for large $x_i^U - x_i^L$

- Many variants
  - piecewise
  - first decompose function
  - exponential function $f(x) - \sum_i (1 - \exp(\gamma_i (x_i - x_i^L)))(1 - \exp(\gamma_i (x_i^U - x_i)))$



Maranas & Floudas JOGO 4.2 (1994): 135-170.
Akrotirianakis & Floudas JOGO 30.4 (2004): 367-390.

McCormick Relaxations: With Auxiliary Variables and in Original Variable Space

Auxiliary variable method

$$\min_{x} \ \exp(x) - x^3 \ \ \text{s.t.} \ x \in [-1{,}1.5]$$

$\boldsymbol{z_1 = \exp(x)}$

$\boldsymbol{z_2 = -x^3}$

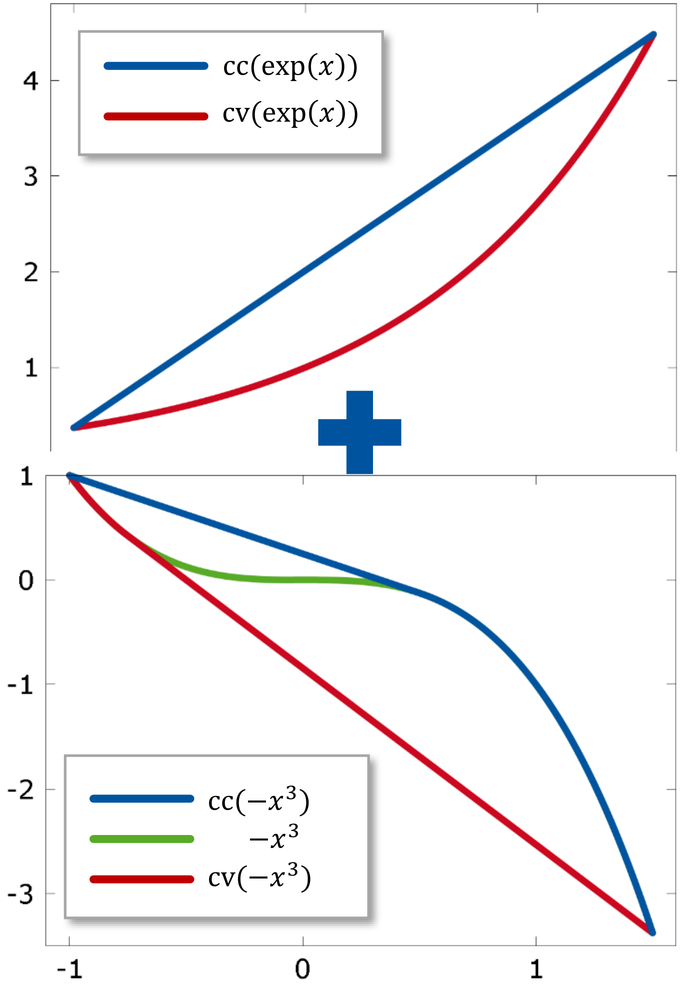
$$\min_{x,z_1,z_2} \ z_1 + z_2 \quad \text{s.t.} \ z_1 = \exp(x), \ z_2 = -x^3, \ x \in [-1{,}1.5]$$

convexify

$$\min_{x,z_1,z_2} \ z_1 + z_2$$
$$\text{s.t.} \ \text{cv}(\exp(x)) \le z_1 \le \text{cc}(\exp(x))$$
$$\text{cv}(-x^3) \le z_2 \le \text{cc}(-x^3)$$
$$x \in [-1{,}1.5]$$

linearize

$$\min_{x,z_1,z_2} \ z_1 + z_2$$
$$\text{s.t.} \ \textbf{lin}(\text{cv}(\exp(x))) \le z_1 \le \textbf{lin}(\text{cc}(\exp(x)))$$
$$\textbf{lin}(\text{cv}(-x^3)) \le z_2 \le \textbf{lin}(\text{cc}(-x^3))$$
$$x \in [-1{,}1.5]$$



Multivariate McCormick[1,2]

$$\min_{x} \ \exp(x) - x^3$$
$$\text{s.t.} \ x \in [-1{,}1.5]$$

convexify

$$\min_{x} \ \text{cv}(\exp(x) - x^3)$$
$$\text{s.t.} \ x \in [-1{,}1.5]$$

[1] McCormick, *Mathematical Programming 10 (1976)*
[2] Tsoukalas & Mitsos, *Journal of Global Optimization 59 (2014)*
[3] Mitsos et al., *SIAM Journal on Optimization 20(2) (2009)*

Aachener Verfahrenstechnik

RWTH AACHEN UNIVERSITY

McCormick Relaxations: With Auxiliary Variables and in Original Variable Space

Auxiliary variable method

$$\min_{x} \ \exp(x) - x^3 \ \text{ s.t. } x \in [-1,1.5]$$

$\boldsymbol{z_1 = \exp(x)}$
$\boldsymbol{z_2 = -x^3}$

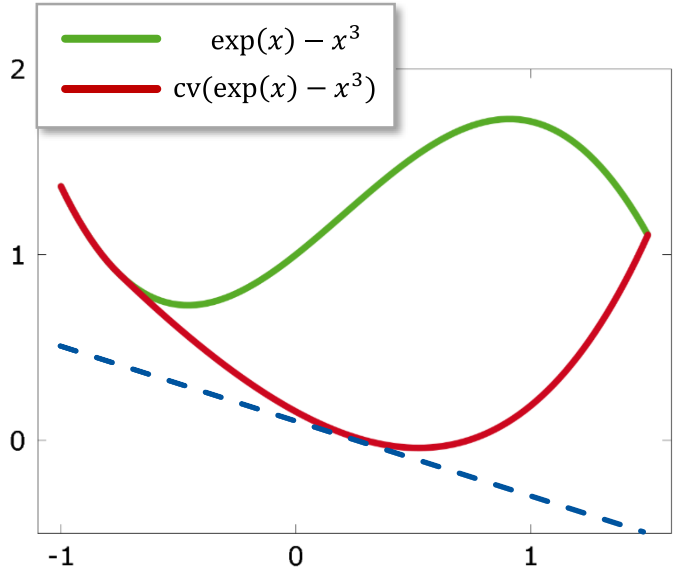
$$\min_{x,z_1,z_2} \ z_1 + z_2 \quad \text{s.t. } z_1 = \exp(x),\ z_2 = -x^3,\ x \in [-1,1.5]$$

convexify

$$\min_{x,z_1,z_2} \ z_1 + z_2$$
$$\text{s.t. } \operatorname{cv}(\exp(x)) \le z_1 \le \operatorname{cc}(\exp(x))$$
$$\operatorname{cv}(-x^3) \le z_2 \le \operatorname{cc}(-x^3)$$
$$x \in [-1,1.5]$$

linearize

$$\min_{x,z_1,z_2} \ z_1 + z_2$$
$$\text{s.t. } \mathbf{lin}(\operatorname{cv}(\exp(x))) \le z_1 \le \mathbf{lin}(\operatorname{cc}(\exp(x)))$$
$$\mathbf{lin}(\operatorname{cv}(-x^3)) \le z_2 \le \mathbf{lin}(\operatorname{cc}(-x^3))$$
$$x \in [-1,1.5]$$



Multivariate McCormick[1,2]

$$\min_{x} \ \exp(x) - x^3$$
$$\text{s.t. } x \in [-1,1.5]$$

convexify

$$\min_{x} \ \operatorname{cv}(\exp(x) - x^3)$$
$$\text{s.t. } x \in [-1,1.5]$$

linearize[3]

$$\min_{x} \ \operatorname{lin}(\operatorname{cv}(\exp(x) - x^3))$$
$$\text{s.t. } x \in [-1,1.5]$$

[1] McCormick, *Mathematical Programming 10 (1976)*
[2] Tsoukalas & Mitsos, *Journal of Global Optimization 59 (2014)*
[3] Mitsos et al., *SIAM Journal on Optimization 20(2) (2009)*

Check Yourself

- Describe methods to obtain underestimating functions. What are the underlying assumptions?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

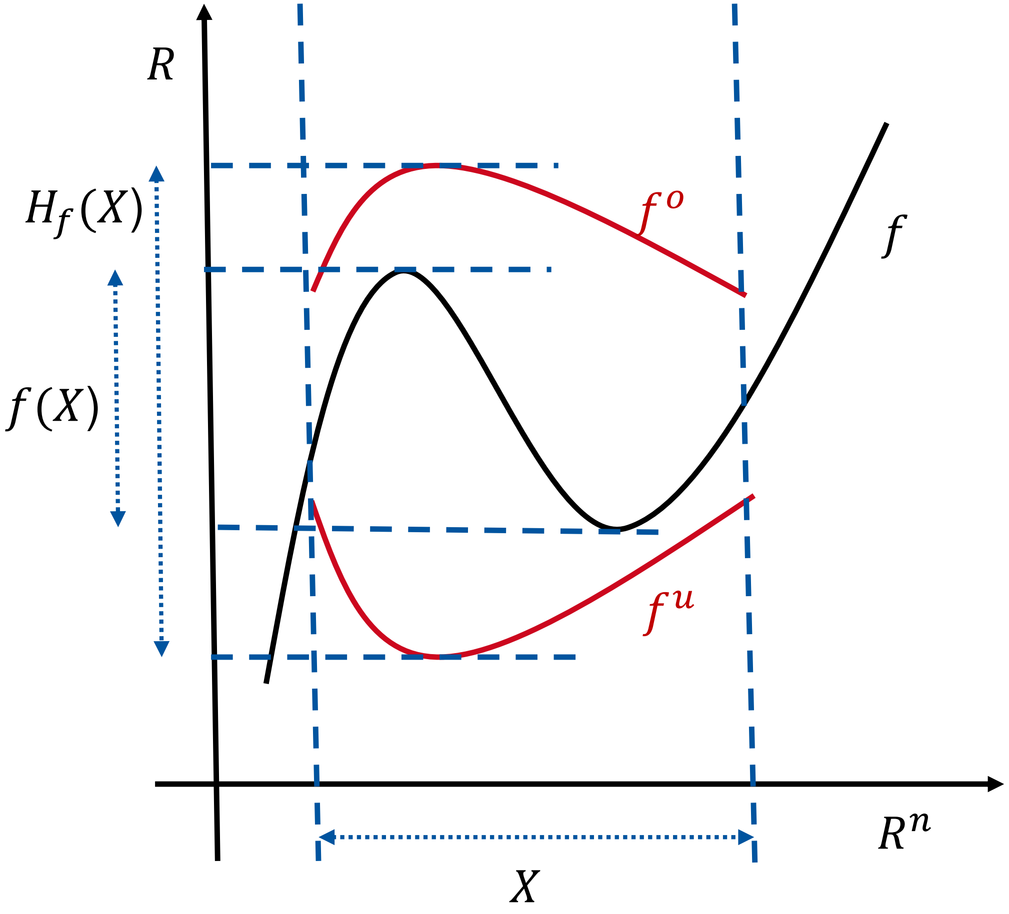
Convergence rate of convex relaxations

Aachener Verfahrenstechnik

RWTH AACHEN UNIVERSITY

Convergence Rate of Relaxations: Theory

- Take $\boldsymbol{x} \in X^0 = [\boldsymbol{x}^{L,0}, \boldsymbol{x}^{U,0}] \supset X = [\boldsymbol{x}^L, \boldsymbol{x}^U]$,
$$\delta(X) = \max_i \{x_i^U - x_i^L\},$$
- Take $f: X^0 \to R$
- We construct pair of relaxations: convex f^u and concave f^o
  - $f^u(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq f^o(\boldsymbol{x}), \forall \boldsymbol{x} \in [\boldsymbol{x}^L, \boldsymbol{x}^U]$
- Tightness desired: small $f(\boldsymbol{x}) - f^u(\boldsymbol{x}), f^o(\boldsymbol{x}) - f(\boldsymbol{x})$
- Convergence: $f^o(\boldsymbol{x}), f^u(\boldsymbol{x}) \to f(\boldsymbol{x})$ for $\delta(X) \to 0$
- Pointwise convergence rate γ: $\exists C > 0$, s.t.,$\forall X \subset X^0$:
$$\sup_{\boldsymbol{x} \in X}\{f(\boldsymbol{x}) - f^u(\boldsymbol{x}), f^o(\boldsymbol{x}) - f(\boldsymbol{x})\} \leq C\big(\delta(X)\big)^\gamma$$
- Hausdorff convergence rate β: $\exists C > 0$, s.t., $\forall X \subset X^0$:
$$\max\{\inf_{\boldsymbol{x} \in X} f(\boldsymbol{x}) - \inf_{\boldsymbol{x} \in X} f^u(\boldsymbol{x}), \sup_{\boldsymbol{x} \in X} f^o(\boldsymbol{x}) - \sup_{\boldsymbol{x} \in X} f(\boldsymbol{x})\}$$
$$\leq C\big(\delta(X)\big)^\beta$$
- Cluster effect: need high convergence rate to avoid creating many nodes in B&B tree
  - < 2 is problematic
  - > 2 is desired
  - $= 2$ is often acceptable

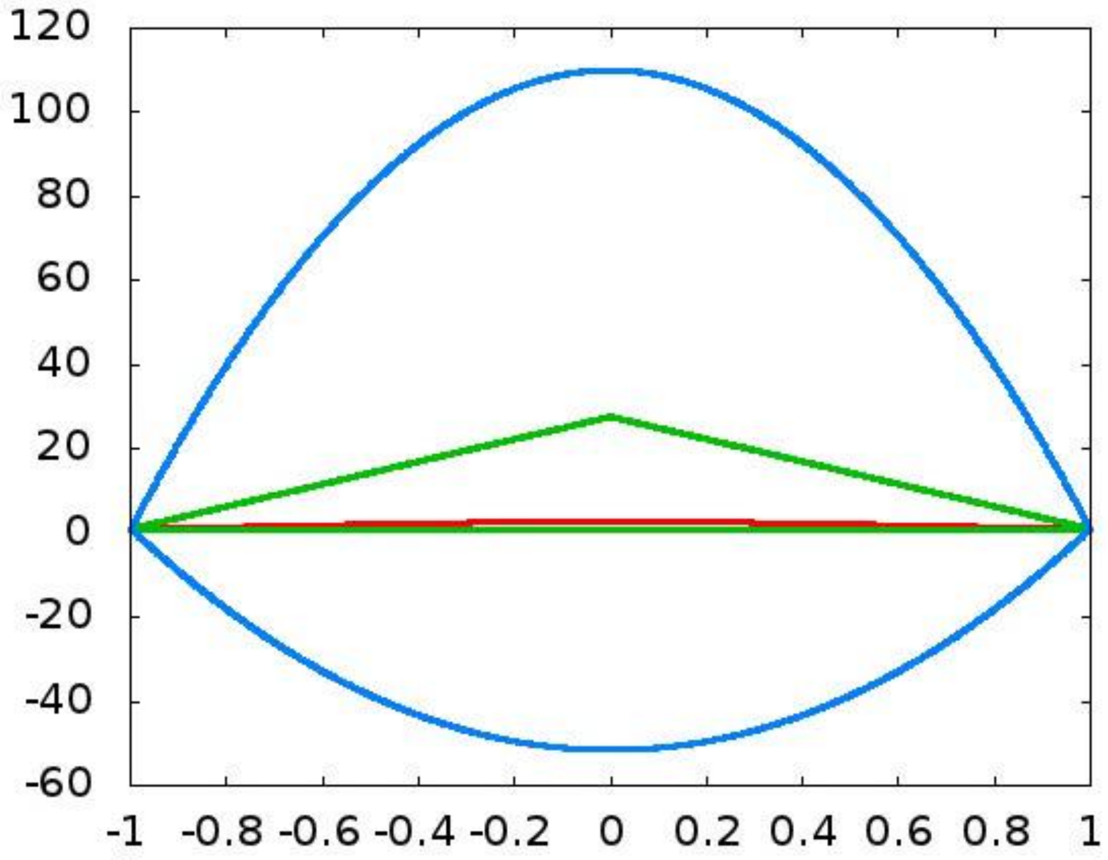


Cluster effect: Du & Kerfott JOGO (1994), Wechsung, Schaber & Barton JOGO (2014)
Convergence rate: Bompadre & Mitsos JOGO (2012), Najman & Mitsos JOGO (2016)

Convergence Rate of Relaxations: Properties (1)

- Two convergence orders: pointwise γ, Hausdorff β
- $\gamma \leq \beta$
- Envelopes of smooth functions have $\gamma = 2$
- $\gamma > 2$ not possible for nonlinear
- Natural interval extensions: $\beta = 1$
  - Other interval extensions: $\beta = 2$
- αBB: $\gamma = 2$, even for fixed α
  - αBB works only for smooth functions
- McCormick: $\gamma = 2$
  - Under mild assumptions
- Relative tightness of different relaxations depends on width of intervals
  Example: $\exp\big((1 - x^2)\big)$
  - red: original function
  - green: original McCormick relaxations with bad decomposition
  - blue: αBB relaxations, with optimized α

Relaxations of $\exp\big((1 + x)(1 - x)\big)$



Convergence rate of McCormick relaxations: Bompadre & Mitsos JOGO (2012)
Convergence rate of multivariate McCormick relaxations: Najman & Mitsos JOGO (2016)

Check Yourself

- What does convergence of relaxations mean? How do we measure the convergence? What convergence properties are established for standard relaxations?

Applied Numerical Optimization

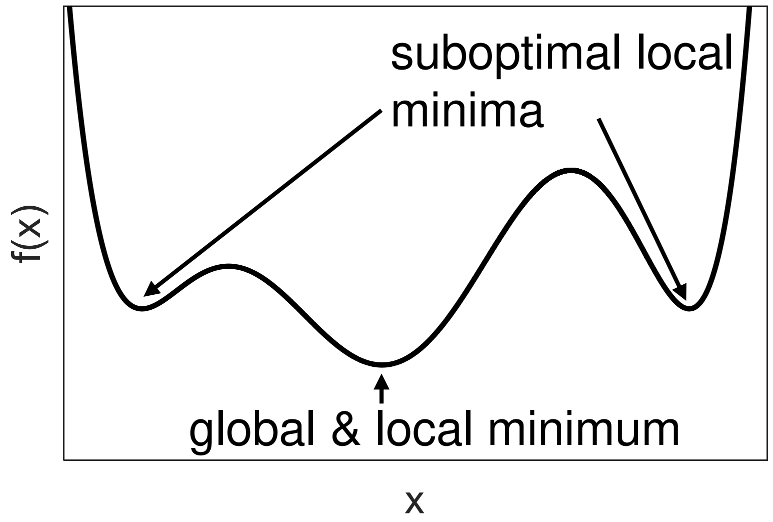
Prof. Alexander Mitsos, Ph.D.

Deterministic global solvers

Aachener Verfahrenstechnik
RWTH AACHEN UNIVERSITY

Global Solution for Nonconvex Problems

- Many engineering/design problems are nonconvex
- Global solution is in principle always desired
  - Sometimes required
  - Sometimes too expensive
  - Sometimes no algorithms exist
- For nonconvex Ω, finding $\boldsymbol{x} \in \Omega$ is a global optimization problem!



Lower Bounds for B&B in Nonconvex Nonlinear Case

- Local methods provide global solution to convex optimization problems
  - In theory, under suitable assumptions
  - In practice there are complications

- Finite bounds required for all variables $\boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U$

- Construct simple underestimations f^u of f on $[\boldsymbol{x}^L, \boldsymbol{x}^U]$
  - $f^u(\boldsymbol{x}) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in [\boldsymbol{x}^L, \boldsymbol{x}^U]$
  - f^u: constant, linear, piecewise linear, convex nonlinear
  - Required: convergence to f as$||\boldsymbol{x}^U - \boldsymbol{x}^L|| \to 0$
  - Desired: tight relaxations and fast convergence rate
  - Active research area

- Treat nonconvex constraints similarly to objective
  - Rewrite equalities as pairs of inequalities
    - $c_i(\boldsymbol{x}, \boldsymbol{y}) = 0$ as $c_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0$ and $-c_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0$
  - Relax inequalities $c_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0$ by underestimating c_i
  - Relax inequalities $c_i(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ by overestimating c_i

Upper Bounds for B&B in Nonconvex Nonlinear Case

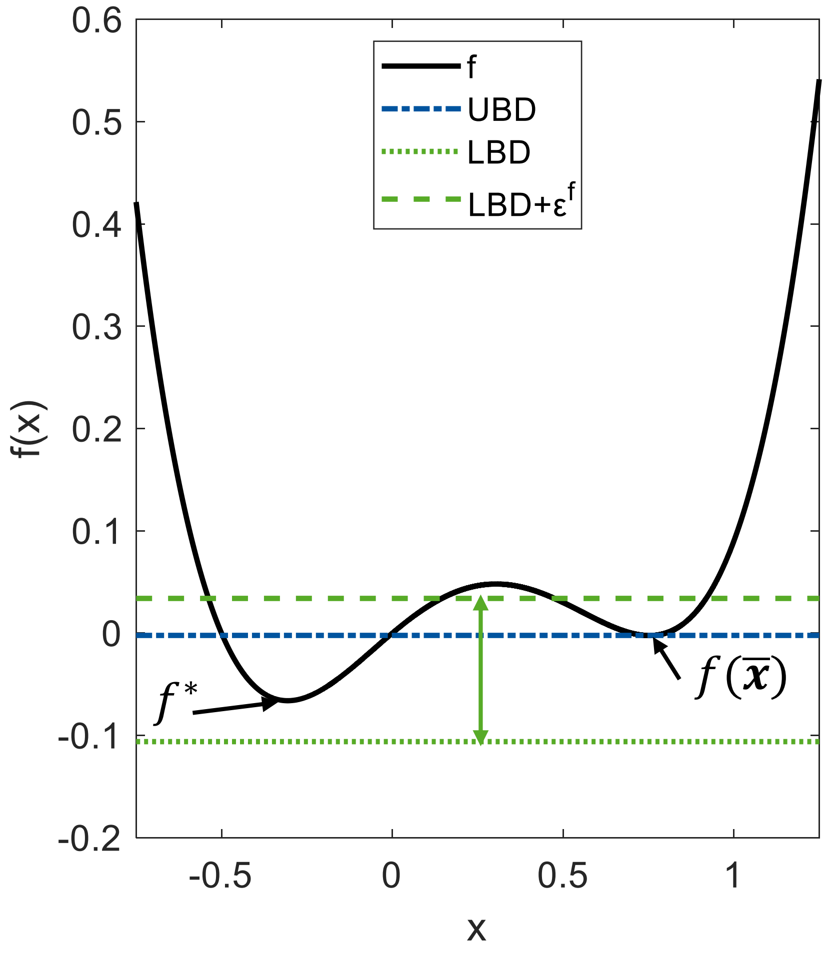
- Any feasible point of NLP suffices as upper bound
  - Better upper bounds give faster convergence
  - Local solution points are desirable
  - Convergence of upper bound is required

- Typically nonconvex restrictions solved locally
  - Restrict continuous variables to smaller ranges
  - Convergence is not trivially satisfied

- Typically upper bound converges quicker than lower bound
  - Proving global optimum more expensive than finding global optimum!
  - Not always true, e.g., for semi-infinite and bilevel optimization

Complications for B&B in Nonconvex Continuous Case

- Branching on continuous variables
  $x_i \in [x_i^L, x_i^U]$ branched to $x_i \in [x_i^L, x_i^M]$ and $x_i \in [x_i^M, x_i^U]$
- Infinite sequence imply that convergence is nontrivial
  - You need to always prove, it is easy to write non-convergent algorithms
- Convergence in the limit to an optimal solution point
- For any user-defined precision ε^f finite termination with
  - $\bar{\boldsymbol{x}} \in \Omega$
  - $LBD \leq f(\boldsymbol{x}^\star) = f^* \leq f(\bar{\boldsymbol{x}})$
  - $UBD = f(\bar{\boldsymbol{x}}) \leq LBD + \varepsilon^f$
  - LBD is a certificate of optimality
  - We do not find $\boldsymbol{x}^\star$ nor f^*, we bound $f^\star$



How to Solve Mixed-Integer Nonlinear Programs (MINLP)?

- MINLPs combine difficulties of NLP and integrality

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{y}} \quad & f(\boldsymbol{x},\boldsymbol{y}) \\
\text{s.t.} \quad & c_i(\boldsymbol{x},\boldsymbol{y}) = 0, \forall\, i \in E \\
& c_i(\boldsymbol{x},\boldsymbol{y}) \leq 0, \forall\, i \in I \\
& \boldsymbol{x} \in R^{n_x}, \text{ continuous} \\
& \boldsymbol{y} \in Y, \text{ discrete (e.g. } \boldsymbol{y} \in \{0,1\}^{n_y})
\end{aligned}
$$

- Branch-and-bound (and do the right thing) is standard method
  - Simple idea: B&B on $\boldsymbol{y}$, globally solve the NLP on each node
  - State of the art: B&B simultaneously on $\boldsymbol{x}$ and $\boldsymbol{y}$, relax nonconvex terms and integrality constraints

- Other global algorithms exist: outer approximation, generalized branch-and-cut, …

- Local solution methods for MINLP exist

Selection of Available Deterministic Global Optimization Solvers

- Antigone (Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations)
  – commercial, developed by Misener in Floudas group
  – decomposition of non-convex constraints and relaxation, auxiliary variables
- BARON (Branch-And-Reduce Optimization Navigator)
  – commercial, developed by Sahinidis group
  – auxiliary variables, first accessible solver
- COUENNE (Convex Over and Under ENvelopes for Nonlinear Estimation)
  – COIN-OR, open source
- EAGO (Easy-Advanced Global Optimization
  – open source, by Stuber group
  – part of JuMp, McCormick relaxations
- LINDO Global
  – commercial
- MAiNGO (McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization)
  – open source, developed by AVT.SVT
  – multivariate McCormick relaxations, reduced space formulations, parallelization
- SCIP (Solving Constraint Integer Programs)
  – free for academic use
  – developed by Vigerske and Gleixner

Check Yourself

- What are the implications of nonconvex objective function?
- What are the implications of nonconvex feasible set?
- Basic assumptions and guarantees of deterministic global algorithms.

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Reduced space for global optimization

Aachener Verfahrenstechnik
RWTH AACHEN UNIVERSITY

Reduced Space vs Full Space Formulation

Full Space (FS)

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; f(\boldsymbol{x},\boldsymbol{z})$$
$$\text{s.t.} \;\; \boldsymbol{c}_I(\boldsymbol{x},\boldsymbol{z}) \le \mathbf{0}$$
$$\boldsymbol{c}_E(\boldsymbol{x},\boldsymbol{z}) = \mathbf{0}$$

Total dim: $\dim(\boldsymbol{x}) + \dim(\boldsymbol{z})$
with $\dim(\boldsymbol{x}) \ll \dim(\boldsymbol{z})$

$\boldsymbol{x}$ degrees of freedom
$\boldsymbol{z}$ state variables

Solve $\boldsymbol{c}_E(\boldsymbol{x},\boldsymbol{z}) = \mathbf{0}$ for $\boldsymbol{z}$

Reduced Space (RS)

$$\min_{\boldsymbol{x}} \; \tilde{f}(\boldsymbol{x})$$
$$\text{s.t.} \;\; \tilde{\boldsymbol{c}}_I(\boldsymbol{x}) \le \mathbf{0}$$

Total dim: $\dim(\boldsymbol{x})$

- NLP

$$\min_{k,\cancel{T_i}} \sum_{i=0}^{n+1} (T_i^m - T_i)^2$$
$$\text{s.t.} \;\; \cancel{\frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} = \frac{q_i^0 + q_i^1 T_i}{k}, i \in \{1, \dots, n+1\}}$$
$$T_0 = 500, T_{n+1} = 600, k \in [0.1, 10], \cancel{T_i \in [0, 2000]}$$

$$\dim(k) + \dim(\mathbf{T})$$
$$1 = \dim(k) \ll \dim(\mathbf{T}) = 99$$

[7] Epperly & Pistikopoulos, JOGO, 11(3), 287-311 (1997) [8] Byrne & Bogle, Ind. Eng. Chem. Res, 39(11), 4296-4301 (2000) [9] Mitsos, Chachuat & Barton, SIOPT, 20(2), 573-601 (2009)
[10] Bongartz, & Mitsos, JOGO, 69(4), 761-796 (2017) [11] Bongartz and Mitsos, JOGO, 69(4), 761-796 (2018)

Reduced Space vs Full Space Formulation

Full Space (FS)

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; f(\boldsymbol{x},\boldsymbol{z})$$
$$\text{s.t.} \;\; \boldsymbol{c}_I(\boldsymbol{x},\boldsymbol{z}) \leq \boldsymbol{0}$$
$$\boldsymbol{c}_E(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{0}$$

Total dim: $\dim(\boldsymbol{x}) + \dim(\boldsymbol{z})$
with $\dim(\boldsymbol{x}) \ll \dim(\boldsymbol{z})$

$\boldsymbol{x}$ degrees of freedom
$\boldsymbol{z}$ state variables

Solve $\boldsymbol{c}_E(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{0}$ for $\boldsymbol{z}$

Reduced Space (RS)

$$\min_{\boldsymbol{x}} \; \tilde{f}(\boldsymbol{x})$$
$$\text{s.t.} \;\; \tilde{\boldsymbol{c}}_I(\boldsymbol{x}) \leq \boldsymbol{0}$$

Total dim: $\dim(\boldsymbol{x})$

- NLP

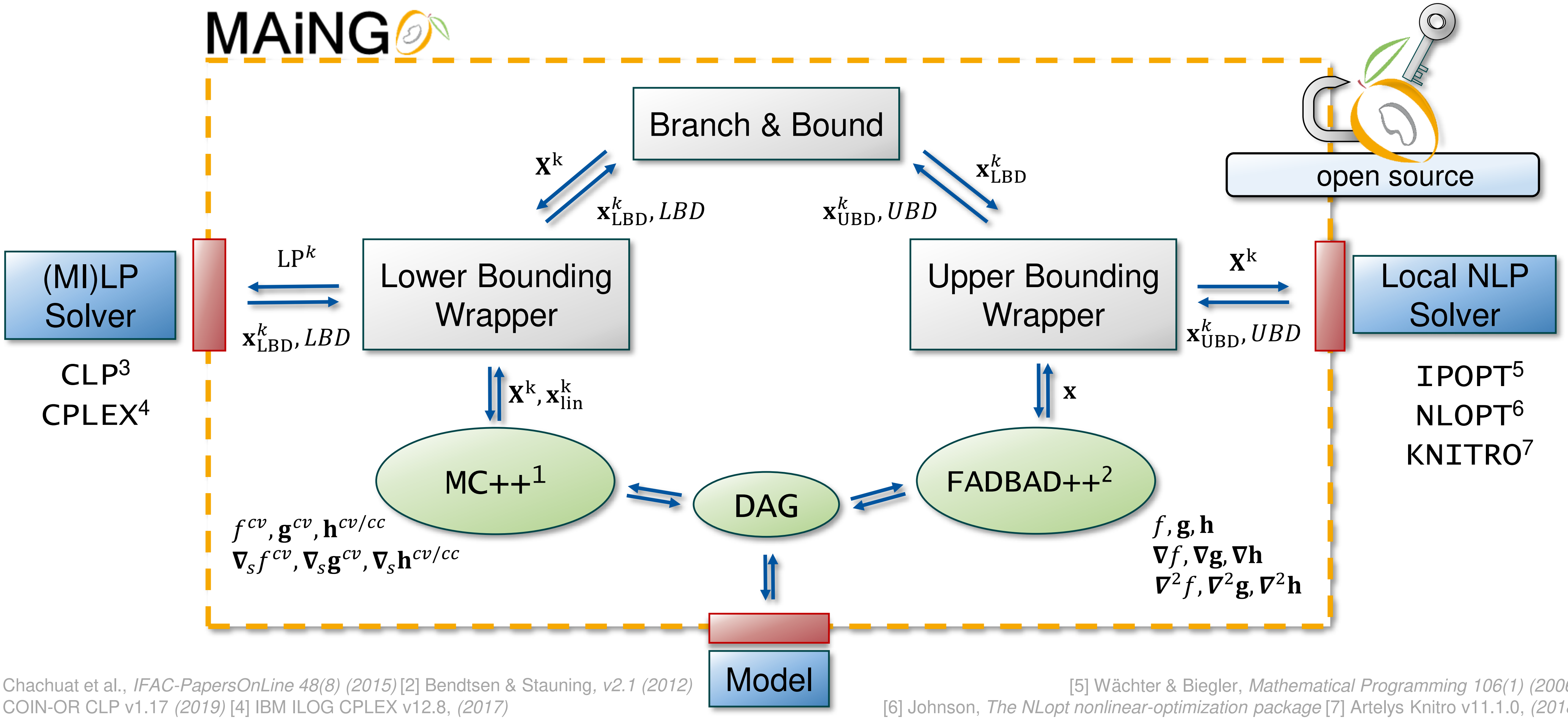
$$\min_{k} \quad \sum_{i=0}^{n+1} (T_i^m - f_i(k))^2$$
$$\text{s.t.} \quad T_0 = 500, T_{n+1} = 600, k \in [0.1,10]$$

$$\dim(k) + \dim(\mathbf{T})$$
$$1 = \dim(k) \ll \dim(\mathbf{T}) = 99$$

$$\begin{pmatrix} 1 & & & & & \\ 1 & \left(-2 - \frac{q_2^1}{k}\Delta x^2\right) & 1 & & & \\ & \cdots & \cdots & \cdots & & \\ & & & 1 & \left(-2 - \frac{q_n^1}{k}\Delta x^2\right) & 1 \\ & & & & & 1 \end{pmatrix} \begin{pmatrix} T_0 \\ T_1 \\ \vdots \\ T_n \\ T_{n+1} \end{pmatrix} = \begin{pmatrix} 500 \\ -\frac{q^0}{k}\Delta x^2 \\ \vdots \\ -\frac{q^0}{k}\Delta x^2 \\ 600 \end{pmatrix}$$

[7] Epperly & Pistikopoulos, JOGO, 11(3), 287-311 (1997) [8] Byrne & Bogle, Ind. Eng. Chem. Res, 39(11), 4296-4301 (2000) [9] Mitsos, Chachuat & Barton, SIOPT, 20(2), 573-601 (2009)
[10] Bongartz, & Mitsos, JOGO, 69(4), 761-796 (2017) [11] Bongartz and Mitsos, JOGO, 69(4), 761-796 (2018)

Implementation Structure of Global Solver MAiNGO



[1] Chachuat et al., *IFAC-PapersOnLine 48(8) (2015)* [2] Bendtsen & Stauning, *v2.1 (2012)*
[3] COIN-OR CLP v1.17 *(2019)* [4] IBM ILOG CPLEX v12.8, *(2017)*
[5] Wächter & Biegler, *Mathematical Programming 106(1) (2006)*
[6] Johnson, *The NLopt nonlinear-optimization package* [7] Artelys Knitro v11.1.0, *(2018)*

https://git.rwth-aachen.de/avt.svt/public/maingo

Check Yourself

- What is the benefit of using the reduced space instead of full space?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Basics of stochastic global optimization

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

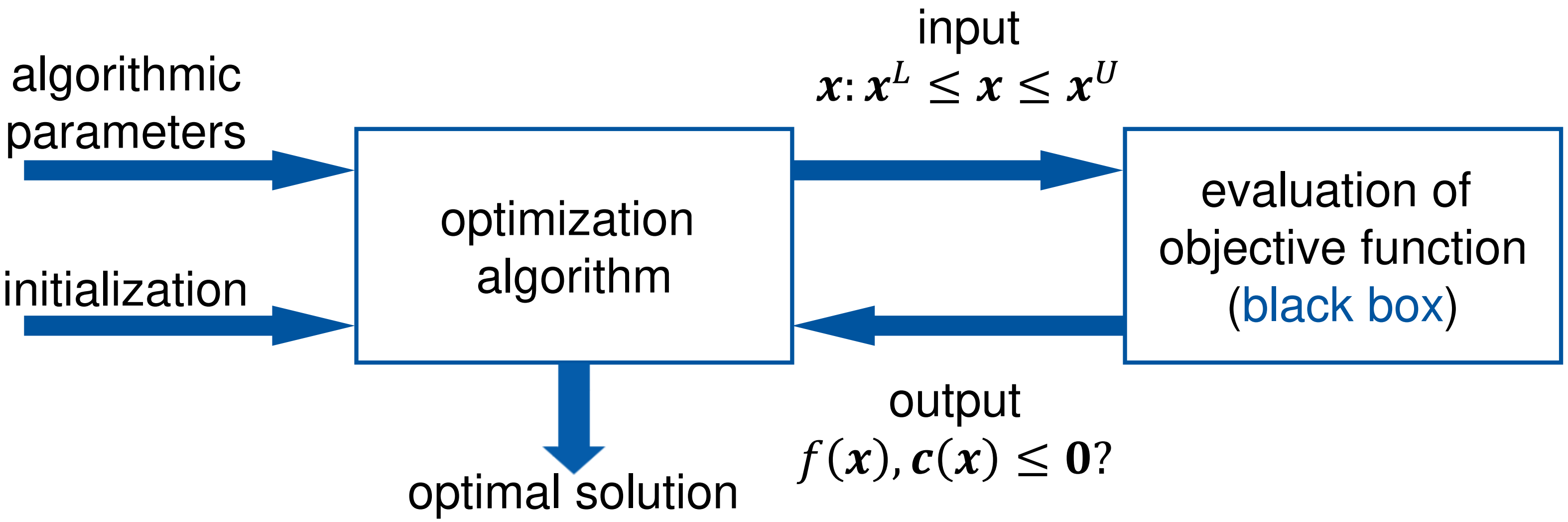
"Black-box" Optimization (alternative meanings exist)

- Only numerical evaluations of functions, no gradients (zero-order oracle):
- Typical formulation

$$\min_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x})$$

$$\Omega = \{\boldsymbol{x} \in R^n | c_i(\boldsymbol{x}) \leq 0, i \in I, \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U\}$$

- Basic idea



Stochastic Global Optimization

- General idea: sample the space.
  - Simple and sophisticated approaches
  - Tradeoff: exploitation vs. exploration
  - Fundamental problem: host set has infinite cardinality

- Promise: avoid getting trapped in suboptimal local solution point
  - Does not avoid exponential complexity
  - As # function evaluations $\rightarrow \infty$, probability of finding a global minimum $\rightarrow 1$

- Advantages: robust, no derivatives required, easy to implement and parallelize, parallelization efficient

- Drawbacks: slower than gradient-based local methods, no rigorous termination criteria, no guarantee to finitely find global optimum/feasible point, no certificate of optimality

- Hybrid methods: combine stochastic with deterministic local solver

- Many methods exist. We describe the basics of popular ideas

No Free-Lunch Theorem

- No-free lunch in everyday life: it is impossible to get something for nothing.
- No-free lunch in economics: cannot make profit without capital and risk of loss
- No-free lunch in stochastic global optimization: any elevated performance of an algorithm for one class of problems is offset by worse performance for another class.
  - True also compared to random search

- Important consequences
  - Comparisons are difficult
  - If possible tune your algorithm to your problems.
    If you have no knowledge about your problem, try many algorithms

Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." *IEEE transactions on evolutionary computation* 1.1 (1997): 67-82.

Random Search

- Random search:
  - starting from initial point $\boldsymbol{x}^{(0)}$
  - randomly choose new iterate $\boldsymbol{x}^{(k+1)}$
  - compare $f(\boldsymbol{x}^{(k+1)})$ and best value found f^*, and update if applicable

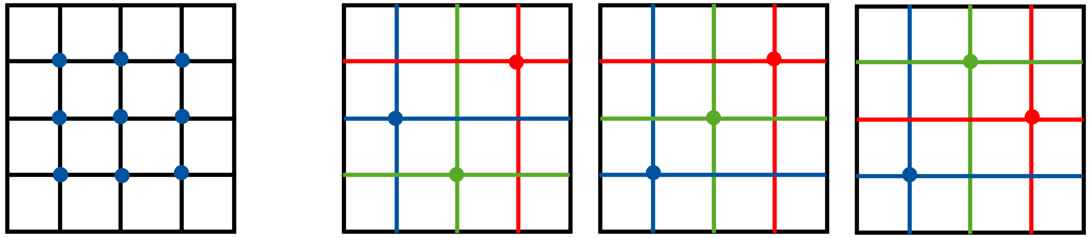
  - ☺ very easy to implement, no special requirements on objective function
  - ☹ requires many function evaluations and provides no guarantee for (finite) convergence

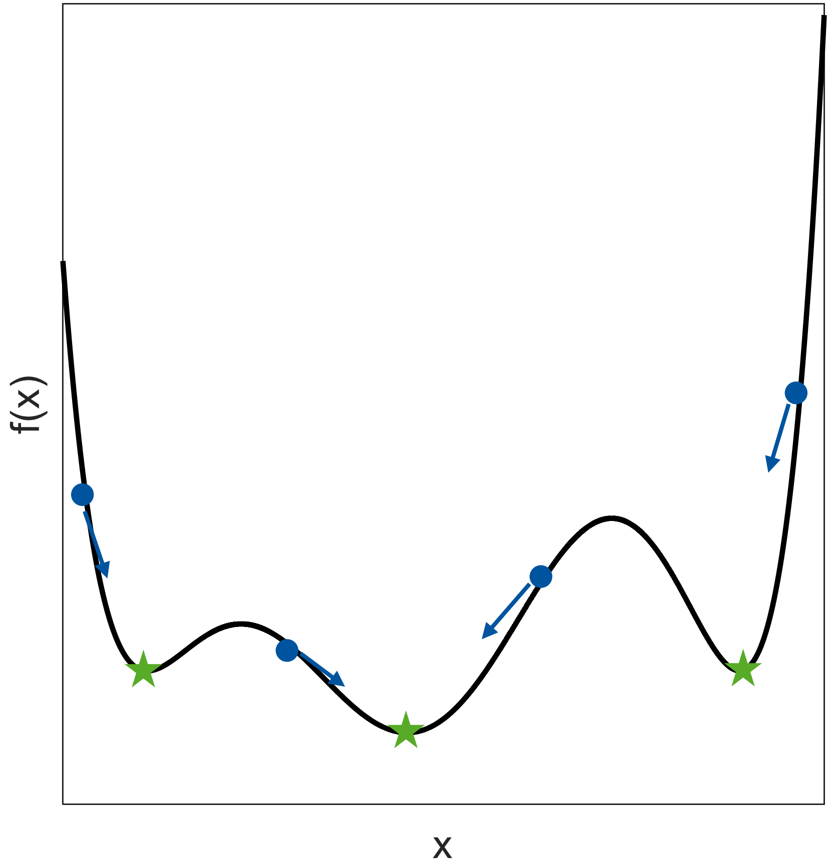
Multistart as a Heuristic for Global Solution of NLPs

- General idea: start local solvers from many initial guesses
  - hope: some will converge to the global minimum
  - by construction hybrid method

- # initial guesses?
  - Theory: we would like to cover space, but this scales exponentially with number of variables
  - In practice: determined by how long you are willing to wait

- Various possibilities to pick initial guesses: grid, latin hypercube, random, physical insight





Practical Recommendations for Multistart

- Think about the problem, try with deterministic solvers
- Parallelize
  - No communication between instances required → submit as separate processes
  - Instances may take long without progress → limit CPU time for each
- Try different solvers simultaneously
  - Possibly repeat same initial guesses with different algorithms
  - Vary solver options for different runs
- Try different formulations
- Record points visited by local solvers to avoid problems with convergence
- Examine pool of solutions
  - Do we have multiple points at the suspected global solution?
  - Are some runs better (e.g. one solver vs another)?

Check Yourself

- Describe random search
- Describe multistart.
- What is the basic idea of stochastic global algorithms? What are their properties, advantages and disadvantages?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

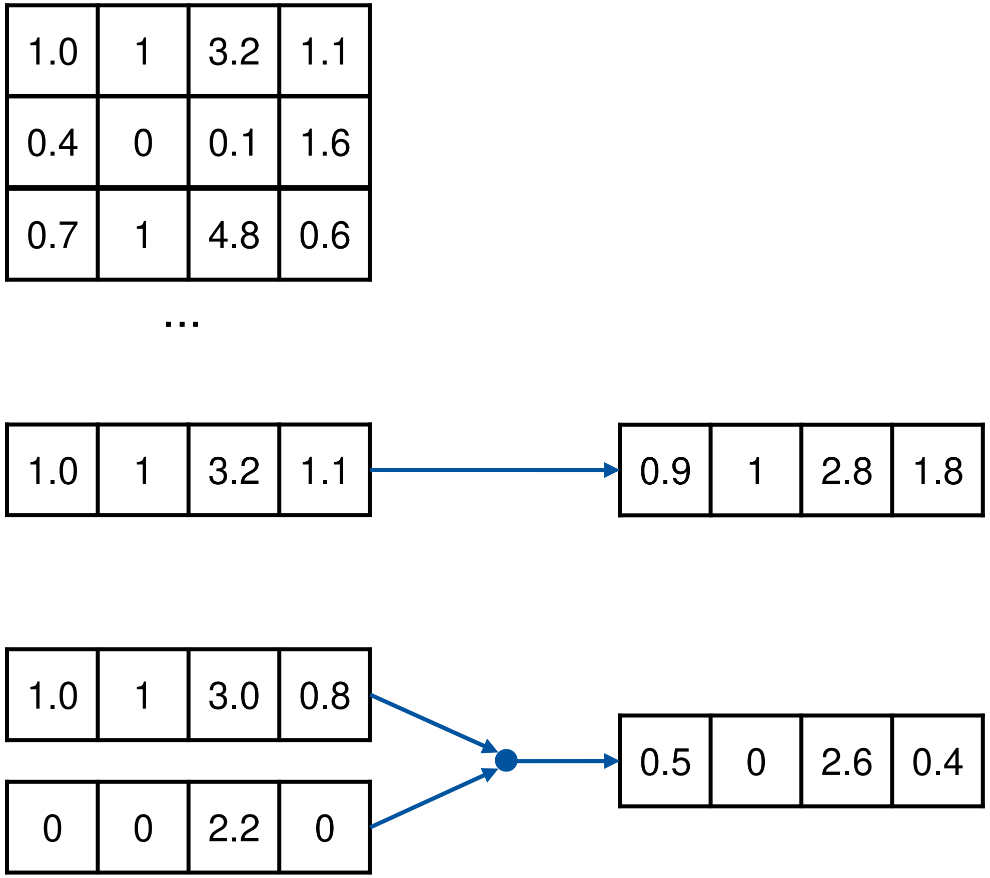
Stochastic global optimization: Genetic Algorithm

Aachener Verfahrenstechnik

RWTH AACHEN UNIVERSITY

Genetic Algorithm: Basic Idea

- Based on simplistic biological principle: survival of the fittest
- Start with an initial population (= initial guesses).
  - At each iteration the size remains fixed
- Accept survivors based on merit function and distance from previous members
  - Merit function: tradeoff of objective and constraints
- Generate new members by mutation: perturb entries randomly
  - Move around in the place, ensures local optimization
- Generate new members by crossing (recombination): child inherits some entries from parents
  - Move far away, avoiding suboptimal points



Genetic Algorithm: Practical Recommendations

- See practical recommendations for multistart algorithms
  - Parallize by MPI or even shell script: manager processor for algorithm, worker processors for function evaluation
- Hybridize with deterministic local solver: run local solver for promising points
- Termination criteria: # iterations, small improvement in objective function
- Plethora of variants → picking best algorithm/solver is hard.
  Alternatives:
  - Take existing solver and tune. Advantage: no need to reinvent the wheel, easy start.
  - Implement basic solver and tune to problem. Advantage: less problems with compatibility (OS, language, license, …), you know pitfalls, you can tailor code to your needs
- Visits many points → can be used to generate pool of solutions
  - All algorithms visit many points, GA hopefully qualitatively different
- Can be easily extended to multiobjective optimization

Check Yourself

- Describe genetic algorithm.
- What is the basic idea of stochastic global algorithms? What are their properties, advantages and disadvantages?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Derivative free optimization

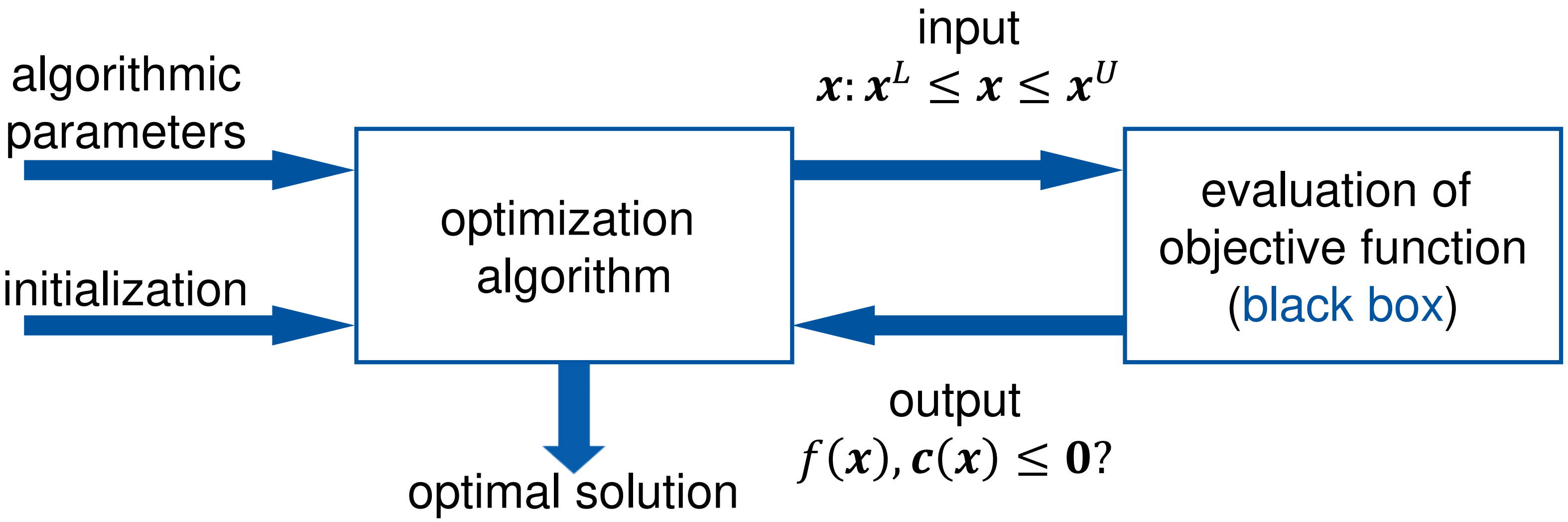
"Black-box" Optimization (alternative meanings exist)

- Only numerical evaluations of functions, no gradients (zero-order oracle):
- Typical formulation

$$\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$$

$$\Omega = \{\boldsymbol{x} \in R^n | c_i(\boldsymbol{x}) \leq 0, i \in I, \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U\}$$

- Basic idea

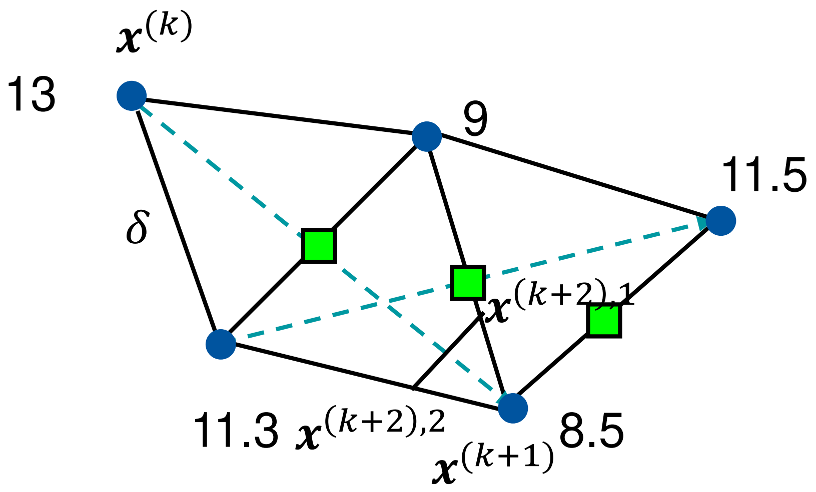


Derivative-free Optimization

- Gradient information may be expensive or not available, e.g.
  - simulation optimization
  - external functions, compiled legacy software
- Gradient evaluation by finite difference prone to errors due to inaccurate function evaluation
- Methods determine new iterate from previous function evaluations
- Non-smoothness does not pose a fundamental problem

Gradient-free Search Methods: Simplex Search

1. choose an initial point $\boldsymbol{x}^{(0)}$ and a $\delta > 0$
2. construct an n-dimensional simplex with edges of length δ, containing $\boldsymbol{x}^{(k)}$ as a vertex
3. evaluate f at each vertex
4. reflect the point with highest value of f to the opposite edge, thus preserving the geometrical shape and define $\boldsymbol{x}^{(k+1)}$.
5. If the procedure does not result in an improvement (close to minimum), reduce the length of the edges and start a new iteration

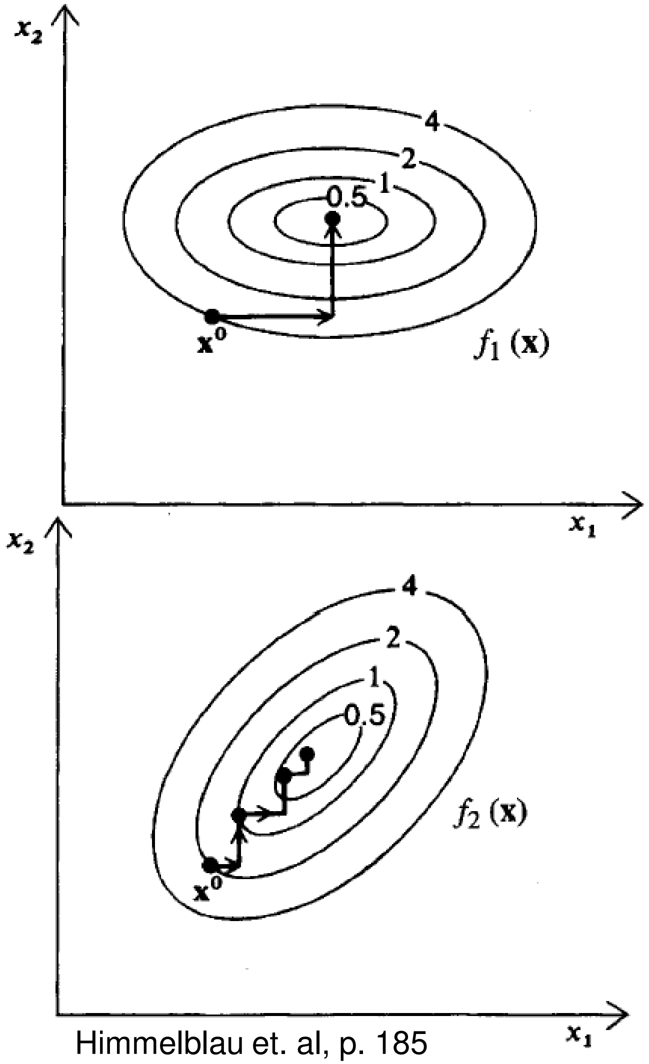


Gradient-free Search Methods: Univariate Search (Coordinate Descent)

Search along the coordinates of the problem.

Basic algorithm:

- choose sequentially component i of $\boldsymbol{x}^{(k)}$ and descend in this direction
- after n steps start from the beginning or reverse the sequence



Himmelblau et. al, p. 185

Check Yourself

- Describe the derivative free methods