Implementation of an αBB-type underestimator in the SGO-algorithm

Anders Skjäl

Process Design & Systems Engineering

November 3, 2010

Refining without branching

- Could the αBB underestimator be used without an explicit branching framework? (cf. the SGO algorithm)

Refining without branching

- Could the αBB underestimator be used without an explicit branching framework? (cf. the SGO algorithm)
- We developed a convex formulation that handles breakpoints with binary variables instead of direct branching

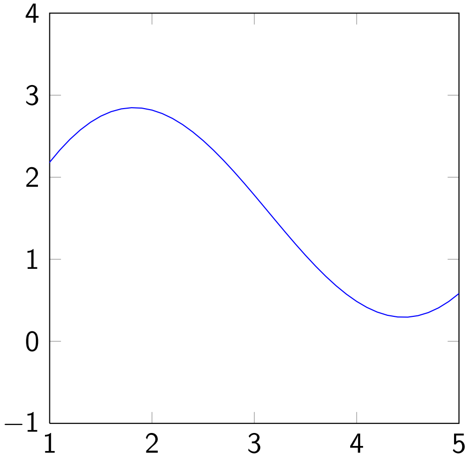
Refining without branching

- Could the αBB underestimator be used without an explicit branching framework? (cf. the SGO algorithm)
- We developed a convex formulation that handles breakpoints with binary variables instead of direct branching
- "Why?"

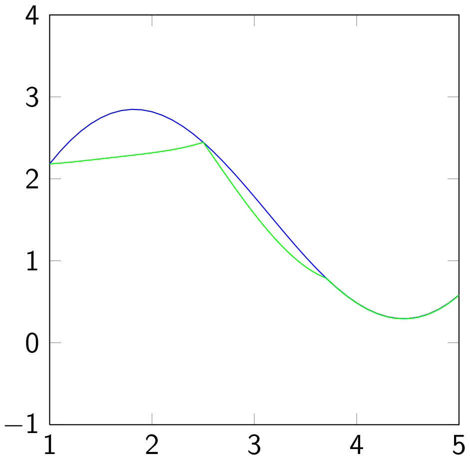
Refining without branching

- Could the αBB underestimator be used without an explicit branching framework? (cf. the SGO algorithm)
- We developed a convex formulation that handles breakpoints with binary variables instead of direct branching
- "Why?"
  - It can readily be integrated with the SGO algorithm
  - It could turn out to be especially well-suited for some types of mixed-integer problems
  - As a convex reformulation it could be of interest in automated reformulation procedures

Refining without branching

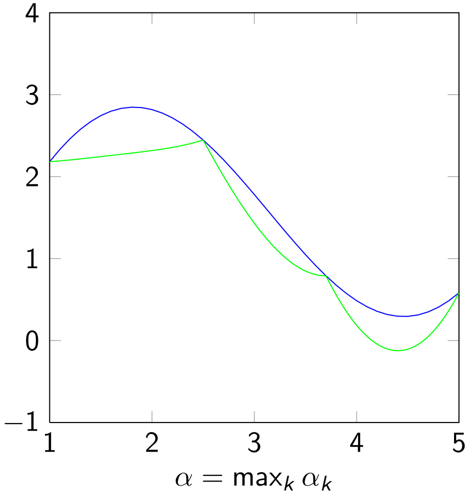


Refining without branching



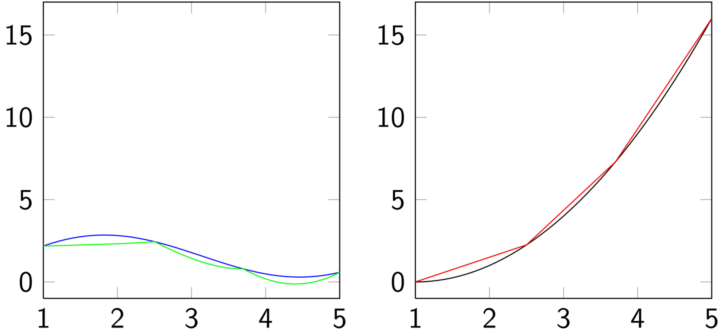
$\alpha_1 = 1, \alpha_2 = 0.6, \alpha_3 = 0$

Refining without branching



Refining without branching

The underestimation error can be described as the difference of a parabola and a piecewise linear function.



Refining without branching

Our formulation: (1D for clarity)

$$f(x) \leq 0 \tag{1}$$

Refining without branching

Our formulation: (1D for clarity)

$$f(x) + \alpha x^2 \leq 0 \tag{1}$$

Refining without branching

Our formulation: (1D for clarity)

$$f(x) + \alpha x^2 - W \leq 0 \tag{1}$$

Refining without branching

Our formulation: (1D for clarity)

$$f(x) + \alpha x^2 - W \leq 0 \tag{1}$$

$$W = \alpha x^2 \tag{2}$$

Refining without branching

Our formulation: (1D for clarity)

$$f(x) + \alpha x^2 - W \leq 0 \quad (1)$$

$$W = \alpha x^2 \quad (2)$$

Overestimating W will relax the feasible domain, we replace W with a piecewise linear function

$$\hat{W} = \sum_{k=1}^{K} A_k b_k + (B_k - A_k) s_k \quad (3)$$

where

$$A_k = \alpha \underline{x_k}^2$$
$$B_k = \alpha \overline{x_k}^2$$

($\underline{x_k}$ and $\overline{x_k}$ denote the interval endpoints)

Refining without branching

The b_k are binary variables, s_k are continuous and nonnegative.

$$\hat{W} = \sum_{k=1}^{K} A_k b_k + (B_k - A_k) s_k$$

Refining without branching

The b_k are binary variables, s_k are continuous and nonnegative.

$$\hat{W} = \sum_{k=1}^{K} A_k b_k + (B_k - A_k) s_k$$

We relate x to b_k and s_k with the constraints

$$\sum_{k=1}^{K} b_k = 1$$

$$s_k \leq b_k, \; \forall k$$

$$x = \sum_{k=1}^{K} \underline{x_k} b_k + (\overline{x_k} - \underline{x_k}) s_k$$

Refining without branching

The b_k are binary variables, s_k are continuous and nonnegative.

$$\hat{W} = \sum_{k=1}^{K} A_k b_k + (B_k - A_k) s_k$$

We relate x to b_k and s_k with the constraints

$$\sum_{k=1}^{K} b_k = 1$$

$$s_k \leq b_k, \ \forall k$$

$$x = \sum_{k=1}^{K} \underline{x_k} b_k + (\overline{x_k} - \underline{x_k}) s_k$$

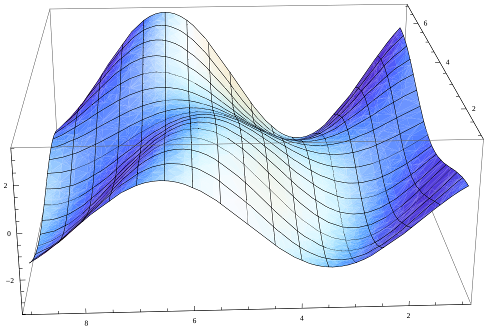
Every constraint is convex and the feasible set is relaxed

In two dimensions

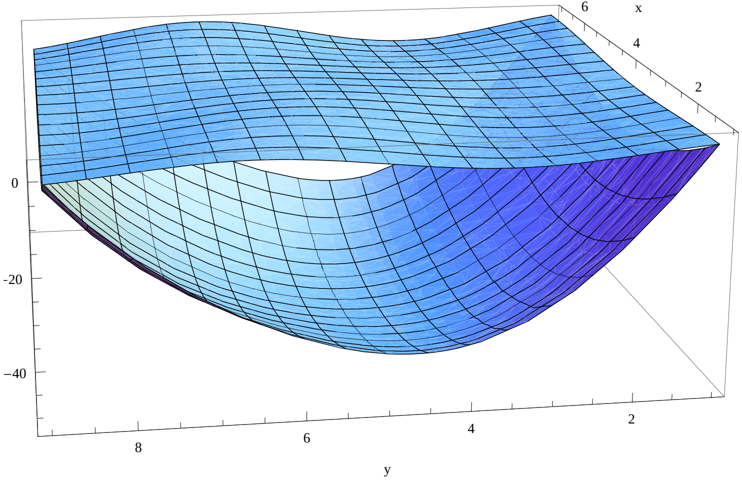
An example:

$$f(x, y) = \sin(x + y) + \sqrt{x} \cos y$$

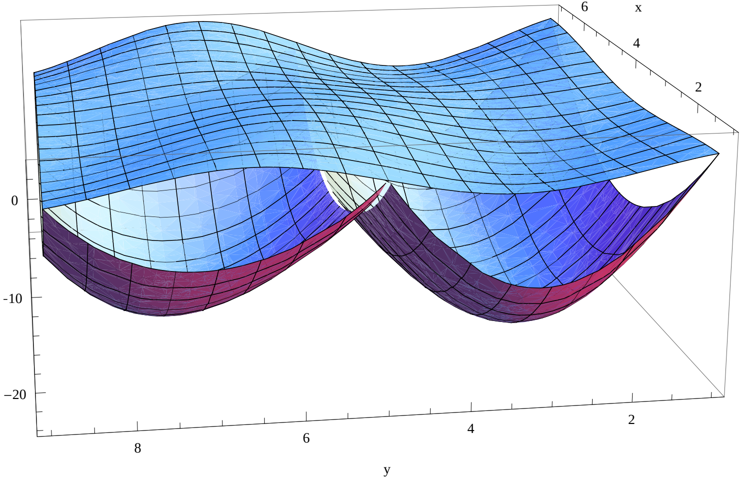
$$1 \leq x \leq 7, \quad 1 \leq y \leq 9$$



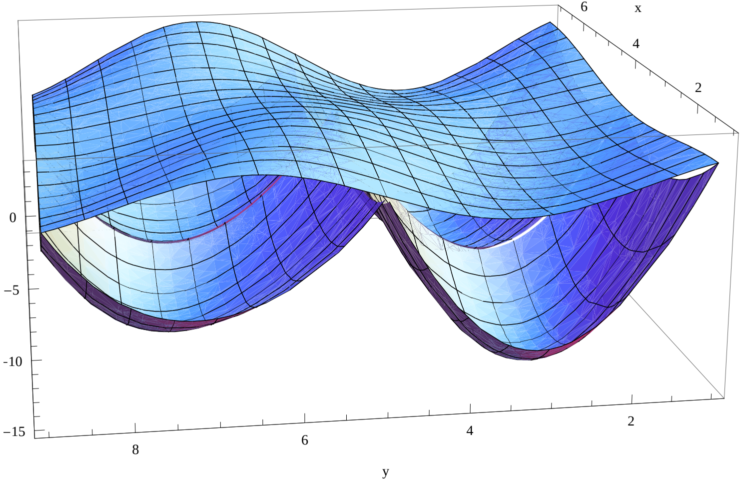
Underestimator - no breakpoints



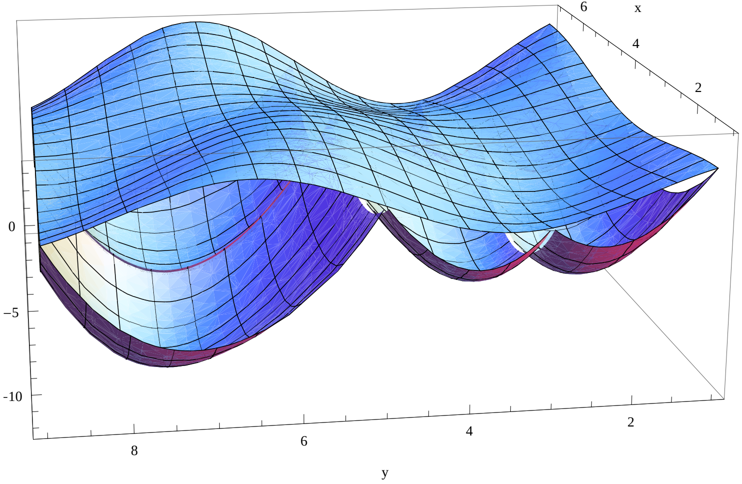
Underestimator - 1 breakpoint



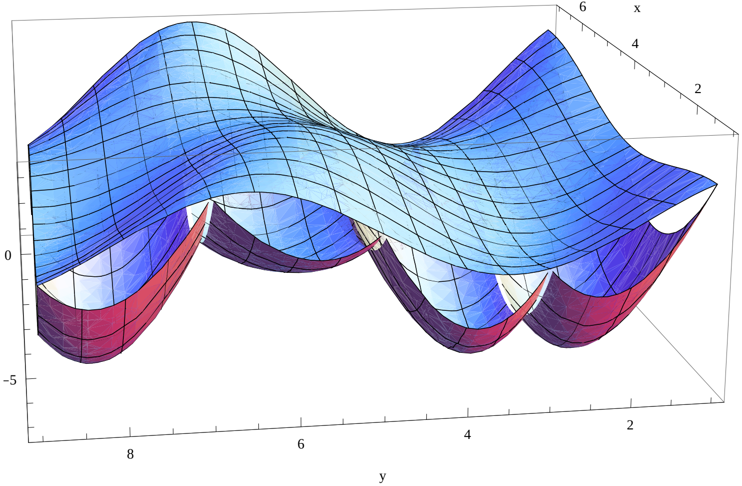
Underestimator - 1+1 breakpoints



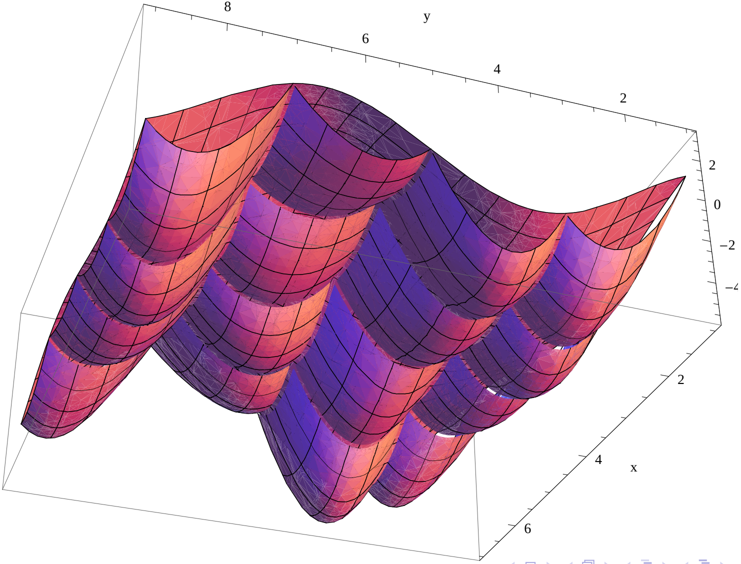
Underestimator - 1+2 breakpoints



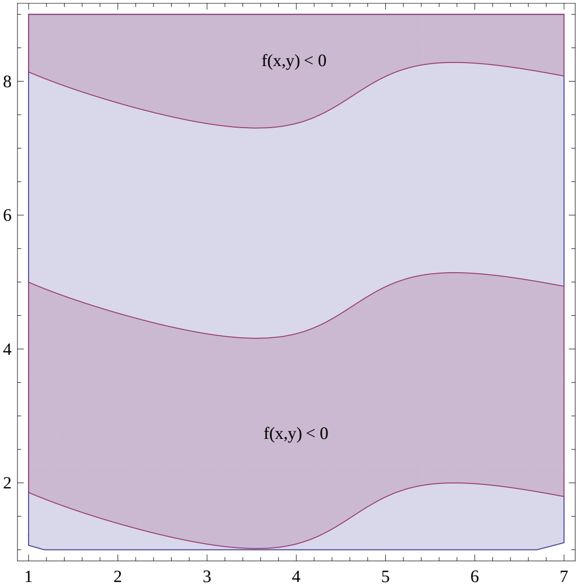
Underestimator - 1+3 breakpoints



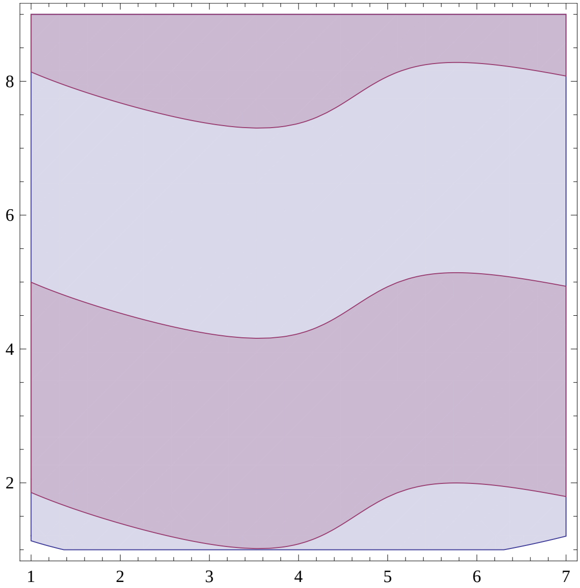
Underestimator - 3+3 breakpoints



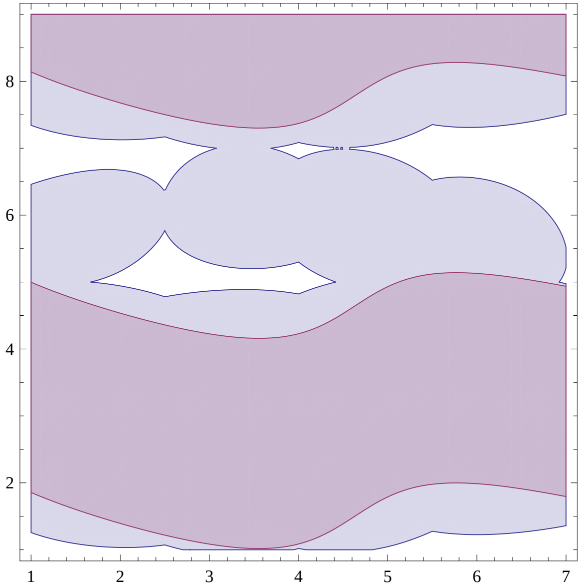
Constraint feasibility



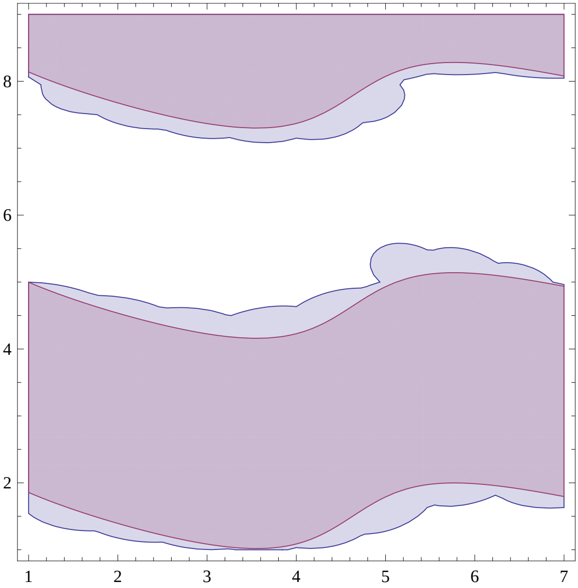
Constraint feasibility - 1+1 breakpoints



Constraint feasibility - 3+3 breakpoints



Constraint feasibility - 7+7 breakpoints



About convergence

The largest underestimation error in a subdomain depends only on the value of $\alpha_i, i = 1, \ldots n$ and the size of the subdomain: (1D)

$$\max_{x \in [\underline{x_k}, \overline{x_k}]} \hat{W} - \alpha x^2 = \max_{x \in [\underline{x_k}, \overline{x_k}]} -\alpha(x - \underline{x_k})(x - \overline{x_k}) = \alpha \left(\frac{\overline{x_k} - \underline{x_k}}{2} \right)^2$$

An ϵ precision is guaranteed if the width of the interval

$$\overline{x_k} - \underline{x_k} \leq \sqrt{\frac{4\epsilon}{\alpha}}.$$

$\Rightarrow$ The algorithm will converge

Challenges & Ideas

- The subproblems grow as we add breakpoints, the branching is "hidden" in the complexity of the convex MINLPs

Challenges & Ideas

- The subproblems grow as we add breakpoints, the branching is "hidden" in the complexity of the convex MINLPs
  - The increase in complexity depends on the number of breakpoints, not directly on the number of constraints

Challenges & Ideas

- The subproblems grow as we add breakpoints, the branching is "hidden" in the complexity of the convex MINLPs
  - The increase in complexity depends on the number of breakpoints, not directly on the number of constraints
- Bound reductions are only partially applicable

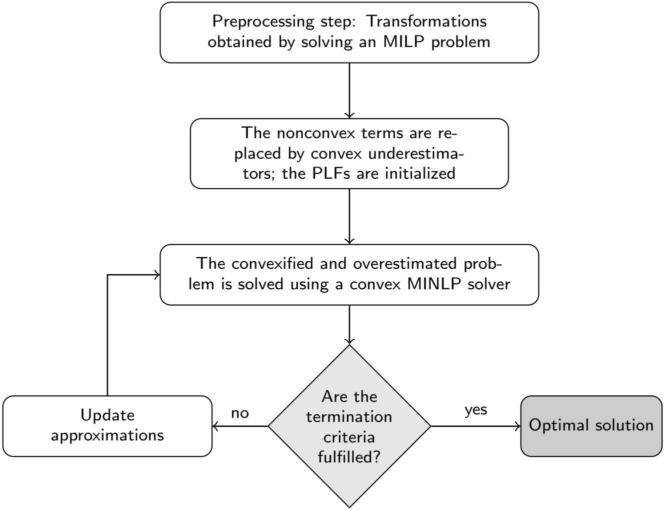
Challenges & Ideas

- The subproblems grow as we add breakpoints, the branching is "hidden" in the complexity of the convex MINLPs
  - The increase in complexity depends on the number of breakpoints, not directly on the number of constraints
- Bound reductions are only partially applicable
- We get less information about subdomains as compared to branch-and-bound

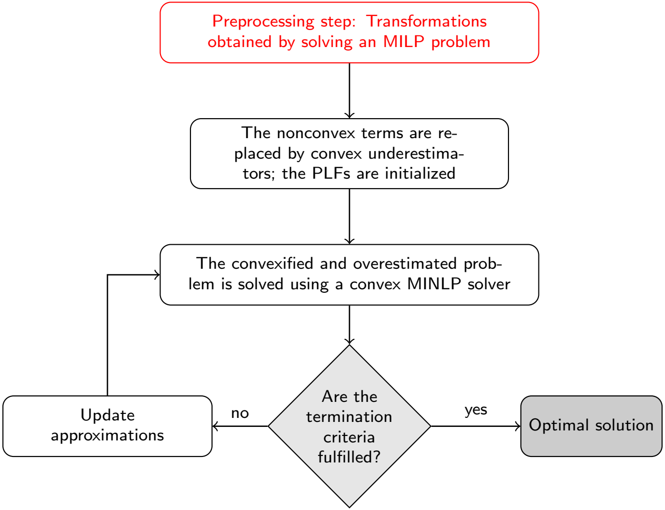
Challenges & Ideas

- The subproblems grow as we add breakpoints, the branching is "hidden" in the complexity of the convex MINLPs
  - The increase in complexity depends on the number of breakpoints, not directly on the number of constraints
- Bound reductions are only partially applicable
- We get less information about subdomains as compared to branch-and-bound
- A type of "minor" breakpoint halving every interval can be introduced without too much cost

Integration with SGO



Integration with SGO



References

C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier.
A global optimization method, αBB, for general twice-differentiable constrained NLPs – I.
Computers & Chemical Engineering, 22(9):1137 – 1158, 1998.

C.A. Floudas.
Deterministic Global Optimization.
Kluwer Academic Publishers, 2000.

A. Lundell, J. Westerlund, and T. Westerlund.
Some transformation techniques with applications in global optimization.
Journal of Global Optimization, 43(2):391–402, 2009.

A. Lundell and T. Westerlund.
Convex underestimation strategies for signomial functions.
Optimization Methods and Software, 24:505–522, 2009.

Thank you

C^2 constraints

- We are interested in handling constraints

$$f(x) \leq 0$$

where $f \in C^2$, i.e. f is twice continuously differentiable.

C^2 constraints

- We are interested in handling constraints

$$f(x) \leq 0$$

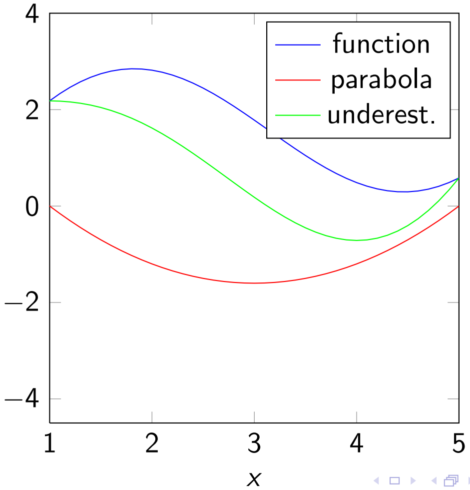
where $f \in C^2$, i.e. f is twice continuously differentiable.

- In addition C^2 objective functions can be handled by rewriting

$$\min \quad f(x) \qquad \text{as} \qquad \begin{array}{ll} \min & \mu \\ \text{s.t.} & f(x) - \mu \leq 0 \end{array}$$

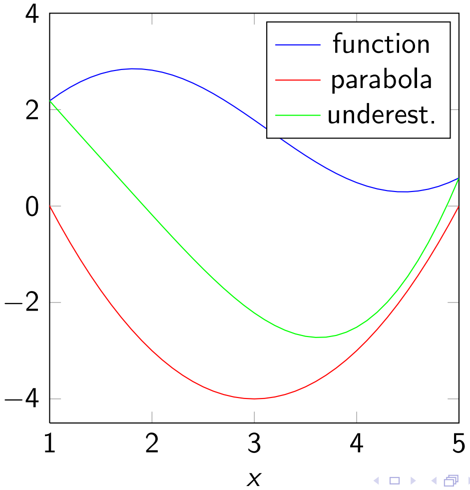
The αBB underestimator

A C^2 function f on the domain $[x_L, x_U] \subset \mathbb{R}$ can always be convexified by adding a parabola $p(x) = \alpha(x - x_L)(x - x_U)$ with a large enough α.



The αBB underestimator

A C^2 function f on the domain $[x_L, x_U] \subset \mathbb{R}$ can always be convexified by adding a parabola $p(x) = \alpha(x - x_L)(x - x_U)$ with a large enough α.



This convex underestimator can be extended to multiple dimensions. Let f be a C^2 function on $\mathbb{R}^n$. For a large enough α the function $g = f + q$ where

$$q(x_1, \ldots, x_n) = \alpha \sum_{i=1}^{n} (x_i - x_i^L)(x_i - x_i^U)$$

is convex.

The αBB underestimator

This convex underestimator can be extended to multiple dimensions. Let f be a C^2 function on $\mathbb{R}^n$. For a large enough α the function $g = f + q$ where

$$q(x_1, \ldots, x_n) = \alpha \sum_{i=1}^{n} (x_i - x_i^L)(x_i - x_i^U)$$

is convex. Tighter underestimators can be found by letting α depend on i

$$q(x_1, \ldots, x_n) = \sum_{i=1}^{n} \alpha_i (x_i - x_i^L)(x_i - x_i^U)$$

The αBB underestimator

How large must we choose α_i?

- One dimension: g is convex if $g'' = f'' + 2\alpha \geq 0$

The αBB underestimator

How large must we choose α_i?

- One dimension: g is convex if $g'' = f'' + 2\alpha \geq 0$
- In general: g is convex if the Hessian matrix H_g is positive semi-definite

The αBB underestimator

How large must we choose α_i?

- One dimension: g is convex if $g'' = f'' + 2\alpha \geq 0$
- In general: g is convex if the Hessian matrix H_g is positive semi-definite

$$H_g = H_f + 2 \cdot \mathrm{diag}(\alpha_i) =$$

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix} + 2 \begin{pmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{pmatrix}$$

The αBB underestimator

How large must we choose α_i?

- One dimension: g is convex if $g'' = f'' + 2\alpha \geq 0$
- In general: g is convex if the Hessian matrix H_g is positive semi-definite

$$H_g = H_f + 2 \cdot \mathrm{diag}(\alpha_i) =$$

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix} + 2 \begin{pmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{pmatrix}$$

- Choose α_i such that the eigenvalues of H_g are nonnegative **on the relevant domain**

The αBB underestimator

- The α_i are calculated (e.g.) by utilizing interval arithmetic and Gershgorin's circle theorem

The αBB underestimator

- The α_i are calculated (e.g.) by utilizing interval arithmetic and Gershgorin's circle theorem
- Smaller valid choices usually exist, but finding the optimal α_i is in general as hard as the optimization problem itself

The αBB underestimator

- The α_i are calculated (e.g.) by utilizing interval arithmetic and Gershgorin's circle theorem
- Smaller valid choices usually exist, but finding the optimal α_i is in general as hard as the optimization problem itself
- Branch-and-bound methods can be used to solve the original problem

Branch-and-bound search

- Branch to split a domain in two and get tighter underestimators

Branch-and-bound search

- Branch to split a domain in two and get tighter underestimators
- Any feasible solution to the original problem gives an upper bound on the optimal objective value

Branch-and-bound search

- Branch to split a domain in two and get tighter underestimators
- Any feasible solution to the original problem gives an upper bound on the optimal objective value
- Any branch with a lower bound greater than the best found upper bound is fathomed (cut)

Branch-and-bound search

- Branch to split a domain in two and get tighter underestimators
- Any feasible solution to the original problem gives an upper bound on the optimal objective value
- Any branch with a lower bound greater than the best found upper bound is fathomed (cut)
- A number of techniques are used to speed up the search, e.g. *bound reduction*