



# Solving nonconvex MINLP by quadratic approximation

Stefan Vigerske

DFG Research Center MATHEON  
*Mathematics for key technologies*





**workshop** = work in progress report



workshop = work in progress report

new MINLP project in Berlin:

- ▷ cooperation of Humboldt University (Römisch, V.) and Zuse Institute (Grötschel, Bley, Gleixner, Koch, Pfetsch)
- ▷ started two months ago
- ▷ <http://www.math.hu-berlin.de/~stefan/B19>

Aim:

- ▷ extend B&B framework SCIP to handle nonlinear constraints



**workshop** = work in progress report

**new MINLP project** in Berlin:

- ▷ cooperation of **Humboldt University** (Römisches Haus) and **Zuse Institute** (Grötschel, Bley, Gleixner, Koch, Pfetsch)
- ▷ started **two months** ago
- ▷ <http://www.math.hu-berlin.de/~stefan/B19>

**Aim:**

- ▷ extend B&B framework **SCIP** to handle nonlinear constraints
- ▷ start with **quadratic problems** (MIQQP)
- ▷ extend step-by-step to other “types” of constraints (“application-driven”)



**workshop** = work in progress report

**new MINLP project** in Berlin:

- ▷ cooperation of **Humboldt University** (Römisch, V.) and **Zuse Institute** (Grötschel, Bley, Gleixner, Koch, Pfetsch)
- ▷ started **two months** ago
- ▷ <http://www.math.hu-berlin.de/~stefan/B19>

**Aim:**

- ▷ extend B&B framework **SCIP** to handle nonlinear constraints
- ▷ start with **quadratic problems** (MIQQP)
- ▷ extend step-by-step to other “types” of constraints (“application-driven”)

**How to start:**



workshop = work in progress report

new MINLP project in Berlin:

- ▷ cooperation of Humboldt University (Römisch, V.) and Zuse Institute (Grötschel, Bley, Gleixner, Koch, Pfetsch)
- ▷ started two months ago
- ▷ <http://www.math.hu-berlin.de/~stefan/B19>

Aim:

- ▷ extend B&B framework SCIP to handle nonlinear constraints
- ▷ start with quadratic problems (MIQQP)
- ▷ extend step-by-step to other “types” of constraints (“application-driven”)

How to start:

- ▷ “steal other people plans”



**workshop** = work in progress report

**new MINLP project** in Berlin:

- ▷ cooperation of **Humboldt University** (Römisch, V.) and **Zuse Institute** (Grötschel, Bley, Gleixner, Koch, Pfetsch)
- ▷ started **two months** ago
- ▷ <http://www.math.hu-berlin.de/~stefan/B19>

**Aim:**

- ▷ extend B&B framework **SCIP** to handle nonlinear constraints
- ▷ start with **quadratic problems** (MIQQP)
- ▷ extend step-by-step to other “types” of constraints (“application-driven”)

**How to start:**

- ▷ “steal other people plans”
- ▷ integrate some techniques from MINLP solver **LaGO**



SCIP is a Branch-Cut-Price framework for

Solving Constraint Integer Programs

- ▷ developed primarily by T. Achterberg at ZIB
- ▷ constraint programming solver with LP relaxation



SCIP is a Branch-Cut-Price framework for

## Solving Constraint Integer Programs

- ▷ developed primarily by T. Achterberg at ZIB
- ▷ constraint programming solver with LP relaxation
- ▷ can be used as very efficient MIP solver
  - strong preprocessing, branching rules, domain propagators, many heuristics (T. Berthold) and separators (K. Wolter)



SCIP is a Branch-Cut-Price framework for

## Solving Constraint Integer Programs

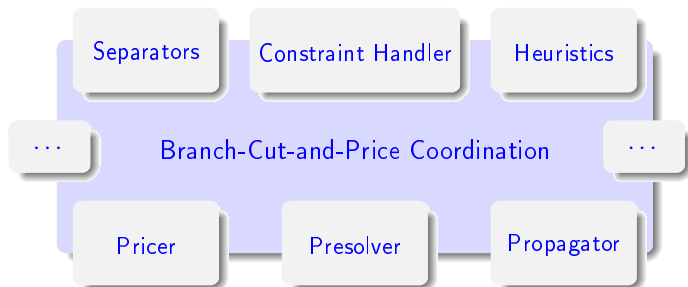
- ▷ developed primarily by T. Achterberg at ZIB
- ▷ constraint programming solver with LP relaxation
- ▷ can be used as very efficient MIP solver
  - strong preprocessing, branching rules, domain propagators, many heuristics (T. Berthold) and separators (K. Wolter)
- ▷ free for academic use, <http://scip.zib.de>



SCIP is a Branch-Cut-Price framework for

## Solving Constraint Integer Programs

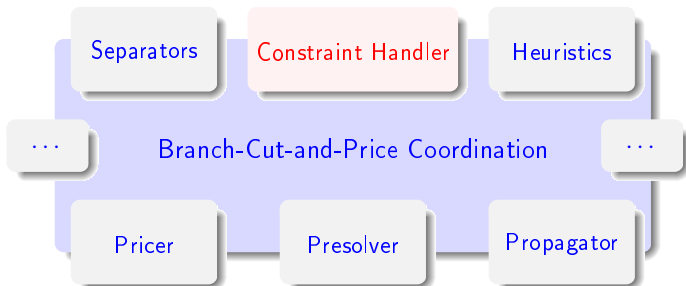
- ▷ developed primarily by T. Achterberg at ZIB
- ▷ constraint programming solver with LP relaxation
- ▷ can be used as very efficient MIP solver
  - strong preprocessing, branching rules, domain propagators, many heuristics (T. Berthold) and separators (K. Wolter)
- ▷ free for academic use, <http://scip.zib.de>
- ▷ constraint oriented; everything is a plugin



SCIP is a Branch-Cut-Price framework for

## Solving Constraint Integer Programs

- ▷ developed primarily by T. Achterberg at ZIB
- ▷ constraint programming solver with LP relaxation
- ▷ can be used as very efficient MIP solver
  - strong preprocessing, branching rules, domain propagators, many heuristics (T. Berthold) and separators (K. Wolter)
- ▷ free for academic use, <http://scip.zib.de>
- ▷ constraint oriented; everything is a plugin





- 1 Introduction
- 2 Constraint Handling**
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule
- 4 Upper Bounds
- 5 Preliminary Numerical Results
- 6 Last Minute Work...
- 7 Conclusions



- 1 Introduction
- 2 **Constraint Handling**
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule
- 4 Upper Bounds
- 5 Preliminary Numerical Results
- 6 Last Minute Work...
- 7 Conclusions



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility

**ENFORCE** checks a LP solution for feasibility and resolves infeasibility by

- ▷ declaring current subproblem infeasible,
- ▷ separate solutions by adding cuts or other constraints,
- ▷ reduce domain of a variable, or
- ▷ branching



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility

**ENFORCE** checks a LP solution for feasibility and resolves infeasibility by

- ▷ declaring current subproblem infeasible,
- ▷ separate solutions by adding cuts or other constraints,
- ▷ reduce domain of a variable, or
- ▷ branching

**LOCK** tell SCIP which variables exist in a constraint and how modifications of variables affect feasibility



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility

**ENFORCE** checks a LP solution for feasibility and resolves infeasibility by

- ▷ declaring current subproblem infeasible,
- ▷ separate solutions by adding cuts or other constraints,
- ▷ reduce domain of a variable, or
- ▷ branching

**LOCK** tell SCIP which variables exist in a constraint and how modifications of variables affect feasibility

**Additional methods** of a constraint handler:

**PRESOLVE** tighten bounds and coefficients, aggregate variables, upgrade constraints to more specific type, ...



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility

**ENFORCE** checks a LP solution for feasibility and resolves infeasibility by

- ▷ declaring current subproblem infeasible,
- ▷ separate solutions by adding cuts or other constraints,
- ▷ reduce domain of a variable, or
- ▷ branching

**LOCK** tell SCIP which variables exist in a constraint and how modifications of variables affect feasibility

**Additional methods** of a constraint handler:

**PRESOLVE** tighten bounds and coefficients, aggregate variables, upgrade constraints to more specific type, ...

**INITLP** generate initial cuts for LP relaxation



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility

**ENFORCE** checks a LP solution for feasibility and resolves infeasibility by

- ▷ declaring current subproblem infeasible,
- ▷ separate solutions by adding cuts or other constraints,
- ▷ reduce domain of a variable, or
- ▷ branching

**LOCK** tell SCIP which variables exist in a constraint and how modifications of variables affect feasibility

**Additional methods** of a constraint handler:

**PRESOLVE** tighten bounds and coefficients, aggregate variables, upgrade constraints to more specific type, ...

**INITLP** generate initial cuts for LP relaxation

**SEPARATE** separate solution from LP relaxation



**Fundamental methods** of a constraint handler:

**CHECK** checks a primal solution for feasibility

**ENFORCE** checks a LP solution for feasibility and resolves infeasibility by

- ▷ declaring current subproblem infeasible,
- ▷ separate solutions by adding cuts or other constraints,
- ▷ reduce domain of a variable, or
- ▷ branching

**LOCK** tell SCIP which variables exist in a constraint and how modifications of variables affect feasibility

**Additional methods** of a constraint handler:

**PRESOLVE** tighten bounds and coefficients, aggregate variables, upgrade constraints to more specific type, ...

**INITLP** generate initial cuts for LP relaxation

**SEPARATE** separate solution from LP relaxation

**PROPAGATE** infer variable domain reductions from current bounds

...



$$\ell \leq x^T A x + b^T x \leq u$$

## PRESOLVE

- ▷ realize fixations and aggregations of variables  
upgrade to linear constraint if possible



$$\ell \leq x^T A x + b^T x \leq u$$

## PRESOLVE

- ▶ realize fixations and aggregations of variables  
upgrade to linear constraint if possible
- ▶ find **block structure**, i.e., partition  $(J_k)_k$  of  $\{1, \dots, n\}$  s.t.:

$$\ell \leq \sum_k x_{J_k}^T A_k x_{J_k} + b^T x \leq u$$



$$\ell \leq x^T A x + b^T x \leq u$$

## PRESOLVE

- ▶ realize fixations and aggregations of variables  
upgrade to linear constraint if possible
- ▶ find **block structure**, i.e., partition  $(J_k)_k$  of  $\{1, \dots, n\}$  s.t.:

$$\ell \leq \sum_k x_{J_k}^T A_k x_{J_k} + b^T x \leq u$$

- ▶ domain reduction



$$\ell \leq x^T A x + b^T x \leq u$$

## PRESOLVE

- ▷ realize fixations and aggregations of variables  
upgrade to linear constraint if possible
- ▷ find **block structure**, i.e., partition  $(J_k)_k$  of  $\{1, \dots, n\}$  s.t.:

$$\ell \leq \sum_k x_{J_k}^T A_k x_{J_k} + b^T x \leq u$$

- ▷ domain reduction
- ▷ compute min/max **eigenvalues** for each  $A_k$



$$\sum_k x_{J_k}^\top A_k x_{J_k} + b^\top x \leq u$$

**SEPARATE, INITLP:** derive linear underestimator for each block



$$\sum_k x_{J_k}^\top A_k x_{J_k} + b^\top x \leq u$$

**SEPARATE, INITLP**: derive linear underestimator for each block

▷ **linearize** if convex (min. eigenvalue  $\lambda_1(A_k) \geq 0$ )



$$\sum_k x_{J_k}^\top A_k x_{J_k} + b^\top x \leq u$$

**SEPARATE, INITLP:** derive linear underestimator for each block

- ▷ linearize if convex (min. eigenvalue  $\lambda_1(A_k) \geq 0$ )
- ▷ apply McCormick for each bilinear term,

$$x_i x_j \geq x_i^L x_j + x_j^L x_i - x_i^L x_j^L$$

$$x_i x_j \geq x_i^U x_j + x_j^U x_i - x_i^U x_j^U$$



$$\sum_k x_{J_k}^\top A_k x_{J_k} + b^\top x \leq u$$

**SEPARATE, INITLP:** derive linear underestimator for each block

- ▷ linearize if convex (min. eigenvalue  $\lambda_1(A_k) \geq 0$ )
- ▷ apply McCormick for each bilinear term,

$$x_i x_j \geq x_i^L x_j + x_j^L x_i - x_i^L x_j^L$$

$$x_i x_j \geq x_i^U x_j + x_j^U x_i - x_i^U x_j^U$$

- ▷ or linearize convex  $\alpha$ -underestimator

$$x_{J_k}^\top A_k x_{J_k} + \lambda_1(A_k)(x_{J_k}^U - x_{J_k})^\top (x_{J_k} - x_{J_k}^L)$$



## PROPAGATE:

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints



## PROPAGATE:

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints

Univariate case:  $ax^2 + bx \in [\ell, u]$

▷ forward propagation: find  $\{ax^2 + bx : x \in [x^L, x^U]\} \cap [\ell, u]$  analytically



## PROPAGATE:

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints

Univariate case:  $ax^2 + bx \in [\ell, u]$

- ▷ forward propagation: find  $\{ax^2 + bx : x \in [x^L, x^U]\} \cap [\ell, u]$  analytically
- ▷ backward propagation: find  $\{x : ax^2 + bx \in [\ell, u]\}$  analytically



## PROPAGATE:

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints

**Univariate case:**  $ax^2 + bx \in [\ell, u]$

- ▷ forward propagation: find  $\{ax^2 + bx : x \in [x^L, x^U]\} \cap [\ell, u]$  analytically
- ▷ backward propagation: find  $\{x : ax^2 + bx \in [\ell, u]\}$  analytically

**Separable case:**  $\sum_k a_k x_k^2 + b_k x_k \in [\ell, u]$

- ▷ forward prop.: find  $\{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}$  analytically and sum up



## PROPAGATE:

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints

**Univariate case:**  $ax^2 + bx \in [l, u]$

- ▷ forward propagation: find  $\{ax^2 + bx : x \in [x^L, x^U]\} \cap [l, u]$  analytically
- ▷ backward propagation: find  $\{x : ax^2 + bx \in [l, u]\}$  analytically

**Separable case:**  $\sum_k a_k x_k^2 + b_k x_k \in [l, u]$

- ▷ forward prop.: find  $\{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}$  analytically and sum up
- ▷ backward prop.: find 
$$\{x_i \mid a_i x_i^2 + b_i x_i \in [l, u] - \sum_{k \neq i} \{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}\}$$

**PROPAGATE:**

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints

**Univariate case:**  $ax^2 + bx \in [\ell, u]$

- ▷ forward propagation: find  $\{ax^2 + bx : x \in [x^L, x^U]\} \cap [\ell, u]$  analytically
- ▷ backward propagation: find  $\{x : ax^2 + bx \in [\ell, u]\}$  analytically

**Separable case:**  $\sum_k a_k x_k^2 + b_k x_k \in [\ell, u]$

- ▷ forward prop.: find  $\{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}$  analytically and sum up
- ▷ backward prop.: find  $\{x_i \mid a_i x_i^2 + b_i x_i \in [\ell, u] - \sum_{k \neq i} \{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}\}$

**Non-separable case:**  $\sum_i a_{i,i} x_i^2 + (b_i + \sum_{j:j \neq i} a_{i,j} x_j) x_i \in [\ell, u]$

- ▷ forward prop.: find  $\{a_{i,i} x_i^2 + (b_i + \sum_{j \neq i} a_{i,j} [x_j^L, x_j^U]) x_i \mid x_i \in [x_i^L, x_i^U]\}$  analytically and sum up

## PROPAGATE:

Domes and Neumaier 2008: Domain Propagation on Quadratic Constraints

**Univariate case:**  $ax^2 + bx \in [\ell, u]$

- ▷ forward propagation: find  $\{ax^2 + bx : x \in [x^L, x^U]\} \cap [\ell, u]$  analytically
- ▷ backward propagation: find  $\{x : ax^2 + bx \in [\ell, u]\}$  analytically

**Separable case:**  $\sum_k a_k x_k^2 + b_k x_k \in [\ell, u]$

- ▷ forward prop.: find  $\{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}$  analytically and sum up
- ▷ backward prop.: find  $\{x_i \mid a_i x_i^2 + b_i x_i \in [\ell, u] - \sum_{k \neq i} \{a_k x_k^2 + b_k x_k \mid x_k \in [x_k^L, x_k^U]\}\}$

**Non-separable case:**  $\sum_i a_{i,i} x_i^2 + (b_i + \sum_{j:j \neq i} a_{i,j} x_j) x_i \in [\ell, u]$

- ▷ forward prop.: find  $\{a_{i,i} x_i^2 + (b_i + \sum_{j \neq i} a_{i,j} [x_j^L, x_j^U]) x_i \mid x_i \in [x_i^L, x_i^U]\}$  analytically and sum up
- ▷ backward prop.: find  $\{x_i \mid a_{i,i} x_i^2 + (b_i + \sum_{j \neq i} a_{i,j} [x_j^L, x_j^U]) x_i \in [\ell, u] - \dots\}$  analytically



- 1 Introduction
- 2 Constraint Handling**
  - Quadratic Constraints (MIQQP)
  - **Nonlinear Constraints (MINLP) - LaGO**
- 3 Branching Rule
- 4 Upper Bounds
- 5 Preliminary Numerical Results
- 6 Last Minute Work...
- 7 Conclusions



## Lagrangian Global Optimizer

Solver for sparse, block-separable, nonconvex MINLPs

2000 Development started by Ivo Nowak as a solver for nonconvex MIQPPs based on Lagrangian decomposition and semidefinite relaxation

2001-2004 extension to MINLP solver

2006 start of COIN-OR project

now Linear-relaxation based Branch and Cut algorithm

Webpage: <https://projects.coin-or.org/LaGO>

Book: Ivo Nowak, Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming, Birkhäuser 2005

Paper: LaGO - a (heuristic) Branch and Cut algorithm for nonconvex MINLPs, Central European Journal of Operations Research 16:2, 2008

**PRESOLVE:**

Given constraint  $h(x) \in [\ell, u]$ ,  $h \in C^2(\mathbb{R}^n, \mathbb{R})$

## PRESOLVE:

Given constraint  $h(x) \in [\ell, u]$ ,  $h \in C^2(\mathbb{R}^n, \mathbb{R})$

▷ sample Hessian to find **block structure** and **quadratic terms**:

$$h(x) = \text{const} + b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r})$$

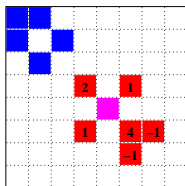
with “small” disjoint subsets  $Q_k$  and  $N_r$  of  $\{1, \dots, n\}$

(TODO: better walk the expression tree)

Example:

$$\begin{aligned} h(x) &= \sin(x_1)x_2 + x_2x_3 \\ &+ x_4^2 + e^{x_5} + x_4x_6 \\ &+ 2x_6^2 - x_6x_7 \end{aligned}$$

$$\nabla^2 h(\cdot) =$$



## PRESOLVE:

Given constraint  $h(x) \in [\ell, u]$ ,  $h \in C^2(\mathbb{R}^n, \mathbb{R})$

- ▷ sample Hessian to find **block structure** and **quadratic terms**:

$$h(x) = \text{const} + b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r})$$

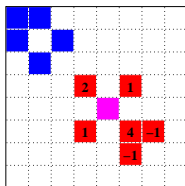
with “small” disjoint subsets  $Q_k$  and  $N_r$  of  $\{1, \dots, n\}$

(TODO: better walk the expression tree)

Example:

$$\begin{aligned} h(x) = & \sin(x_1)x_2 + x_2x_3 \\ & + x_4^2 + e^{x_5} + x_4x_6 \\ & + 2x_6^2 - x_6x_7 \end{aligned}$$

$$\nabla^2 h(\cdot) =$$



- ▷ Check for **convexity/concavity** of each block:  
compute sign of eigenvalues of Hessian in **sample points** (TODO: better)

## PRESOLVE:

Given constraint  $h(x) \in [\ell, u]$ ,  $h \in C^2(\mathbb{R}^n, \mathbb{R})$

- ▷ sample Hessian to find **block structure** and **quadratic terms**:

$$h(x) = \text{const} + b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r})$$

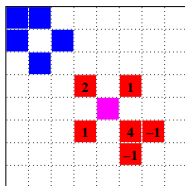
with “small” disjoint subsets  $Q_k$  and  $N_r$  of  $\{1, \dots, n\}$

(TODO: better walk the expression tree)

Example:

$$\begin{aligned} h(x) = & \sin(x_1)x_2 + x_2x_3 \\ & + x_4^2 + e^{x_5} + x_4x_6 \\ & + 2x_6^2 - x_6x_7 \end{aligned}$$

$$\nabla^2 h(\cdot) =$$



- ▷ Check for **convexity/concavity** of each block:  
compute sign of eigenvalues of Hessian in **sample points** (TODO: better)
- ▷ Compute **quadratic under-/overestimators** for each nonconvex/nonconcave  $g_r(x_{N_r})$



Let  $g \in C^2([x^L, x^U], \mathbb{R})$  be nonquadratic and nonconvex.

Compute underestimator

$$q(x) = x^T Ax + b^T x + c$$

of  $g(x)$  by solving the LP

$$\begin{aligned} \min_{A, b, c} \quad & \sum_{x \in S} g(x) - q(x) \\ \text{such that} \quad & q(x) \leq g(x), \quad x \in S, \\ & q(\hat{x}) = g(\hat{x}) \end{aligned}$$

for a **sample set**  $S \subseteq [x^L, x^U]$  and a **reference point**  $\hat{x}$ .



Let  $g \in C^2([x^L, x^U], \mathbb{R})$  be nonquadratic and nonconvex.

Compute underestimator

$$q(x) = x^T Ax + b^T x + c$$

of  $g(x)$  by solving the LP

$$\begin{aligned} \min_{A, b, c} \quad & \sum_{x \in S} g(x) - q(x) \\ \text{such that} \quad & q(x) \leq g(x), \quad x \in S, \\ & q(\hat{x}) = g(\hat{x}) \end{aligned}$$

for a **sample set**  $S \subseteq [x^L, x^U]$  and a **reference point**  $\hat{x}$ .

▷ can be nonconvex



Let  $g \in C^2([x^L, x^U], \mathbb{R})$  be nonquadratic and nonconvex.

Compute underestimator

$$q(x) = x^T Ax + b^T x + c$$

of  $g(x)$  by solving the LP

$$\begin{aligned} \min_{A,b,c} \quad & \sum_{x \in S} g(x) - q(x) \\ \text{such that} \quad & q(x) \leq g(x), \quad x \in S, \\ & q(\hat{x}) = g(\hat{x}) \end{aligned}$$

for a **sample set**  $S \subseteq [x^L, x^U]$  and a **reference point**  $\hat{x}$ .

- ▷ can be nonconvex
- ▷ Quality of  $q(x)$  **depends strongly on the choice of the sample set  $S$**



Let  $g \in C^2([x^L, x^U], \mathbb{R})$  be nonquadratic and nonconvex.

Compute underestimator

$$q(x) = x^T Ax + b^T x + c$$

of  $g(x)$  by solving the LP

$$\begin{aligned} \min_{A,b,c} \quad & \sum_{x \in S} g(x) - q(x) \\ \text{such that} \quad & q(x) \leq g(x), \quad x \in S, \\ & q(\hat{x}) = g(\hat{x}) \end{aligned}$$

for a **sample set**  $S \subseteq [x^L, x^U]$  and a **reference point**  $\hat{x}$ .

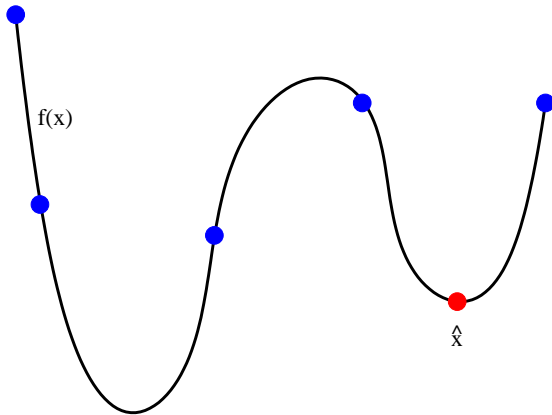
▷ can be nonconvex

▷ Quality of  $q(x)$  **depends strongly on the choice of the sample set**  $S$

⇒ A. Neumaier 2006: adaptive choice of  $S$

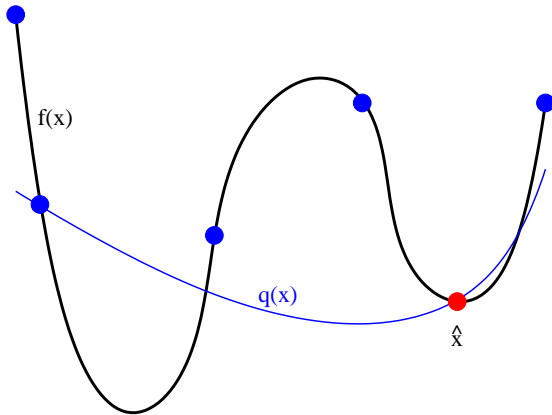


- ▷ initial choice:  $S = \text{vert}([x^L, x^U]) \cup \{x_{\min}, \frac{1}{2}(x^L + x^U)\} \cup M$  with  $\hat{x} := x_{\min}$  a **local minimum** of  $g(x)$  and  $M$  a set of **random points**



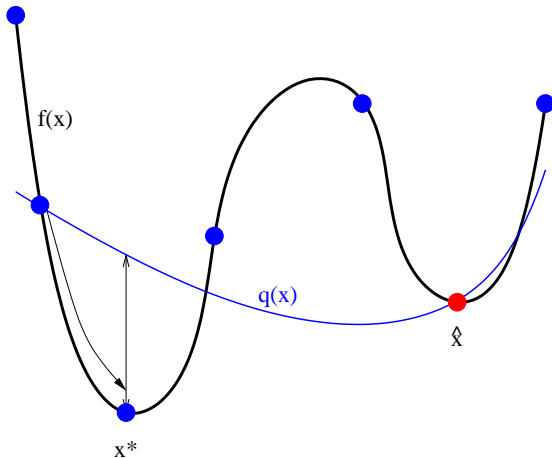


- ▷ initial choice:  $S = \text{vert}([x^L, x^U]) \cup \{x_{\min}, \frac{1}{2}(x^L + x^U)\} \cup M$  with  $\hat{x} := x_{\min}$  a **local minimum** of  $g(x)$  and  $M$  a set of **random points**



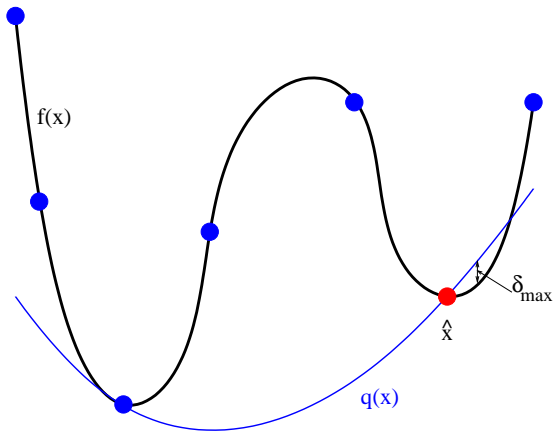


- ▷ for  $x \in S$  with  $g(x) = q(x)$ , maximize the error  $q(x) - g(x) \Rightarrow x^*$
- ▷ if  $q(x^*) - g(x^*) > \delta_{\text{tol}}$ , add  $x^*$  to  $S$  and recompute  $q(x)$



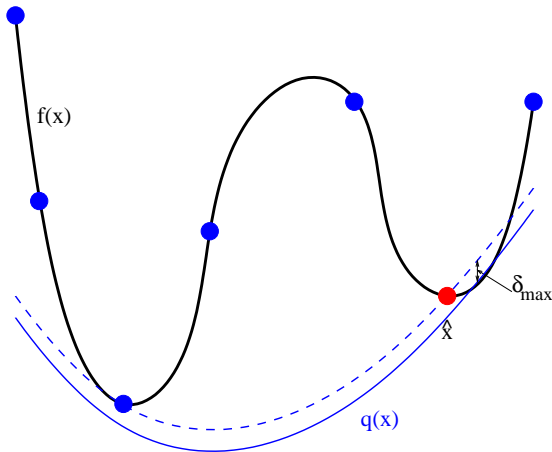


- ▷ for  $x \in S$  with  $g(x) = q(x)$ , maximize the error  $q(x) - g(x) \Rightarrow \delta_{\max}$





- ▷ for  $x \in S$  with  $g(x) = q(x)$ , maximize the error  $q(x) - g(x) \Rightarrow \delta_{\max}$
- ▷ if  $\delta_{\max} < \delta_{\text{tol}}$ , lower  $q(x)$  by  $\delta_{\max}$

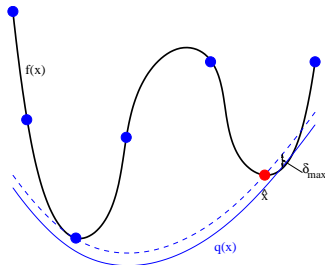




- ▷ still a **heuristic**
- ▷ **expensive** to compute



- ▷ still a **heuristic**
- ▷ **expensive** to compute
- ▷ derived directly from function values  
⇒ can be **tight**
- ▷ can be used as kind of “quadratic cut”  
for **separation**





Consider constraint

$$h(x) := b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r}) \leq u$$

and reference point  $\hat{x}$ ,  $h(\hat{x}) > u$ .



Consider constraint

$$h(x) := b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r}) \leq u$$

and reference point  $\hat{x}$ ,  $h(\hat{x}) > u$ .

SEPARATE, INITLP:

1. for nonconvex  $g_r$ , select quadratic underestimator with highest  $\hat{x}_{N_r}^\top A_r \hat{x}_{N_r}$

$$\Rightarrow \tilde{b}^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_{r: g_r \text{ nonconvex}} x_{N_r}^\top A_r x_{N_r} + \sum_{r: g_r \text{ convex}} g_r(x_{N_r}) \leq \tilde{u}$$



Consider constraint

$$h(x) := b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r}) \leq u$$

and reference point  $\hat{x}$ ,  $h(\hat{x}) > u$ .

SEPARATE, INITLP:

1. for nonconvex  $g_r$ , select quadratic underestimator with highest  $\hat{x}_{N_r}^\top A_r \hat{x}_{N_r}$

$$\Rightarrow \tilde{b}^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_{r: g_r \text{ nonconvex}} x_{N_r}^\top A_r x_{N_r} + \sum_{r: g_r \text{ convex}} g_r(x_{N_r}) \leq \tilde{u}$$

2. linearize convex  $g_r(x_{N_r})$ , linearize  $x_{N_r}^\top A_r x_{N_r}$  as in quad. constraints
3. if not separated, try interval gradient cut



Consider constraint

$$h(x) := b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r}) \leq u$$

and reference point  $\hat{x}$ ,  $h(\hat{x}) > u$ .

**SEPARATE, INITLP:**

1. for nonconvex  $g_r$ , select quadratic underestimator with highest  $\hat{x}_{N_r}^\top A_r \hat{x}_{N_r}$

$$\Rightarrow \tilde{b}^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_{r: g_r \text{ nonconvex}} x_{N_r}^\top A_r x_{N_r} + \sum_{r: g_r \text{ convex}} g_r(x_{N_r}) \leq \tilde{u}$$

2. linearize convex  $g_r(x_{N_r})$ , linearize  $x_{N_r}^\top A_r x_{N_r}$  as in quad. constraints
3. if not separated, try interval gradient cut

**ENFORCE:**

4. if have linearization gap in  $x_{N_r}^\top A_r x_{N_r}$ , do branching



Consider constraint

$$h(x) := b^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_r g_r(x_{N_r}) \leq u$$

and reference point  $\hat{x}$ ,  $h(\hat{x}) > u$ .

**SEPARATE, INITLP:**

1. for nonconvex  $g_r$ , select quadratic underestimator with highest  $\hat{x}_{N_r}^\top A_r \hat{x}_{N_r}$

$$\Rightarrow \tilde{b}^\top x + \sum_k x_{Q_k}^\top A_k x_{Q_k} + \sum_{r: g_r \text{ nonconvex}} x_{N_r}^\top A_r x_{N_r} + \sum_{r: g_r \text{ convex}} g_r(x_{N_r}) \leq \tilde{u}$$

2. linearize convex  $g_r(x_{N_r})$ , linearize  $x_{N_r}^\top A_r x_{N_r}$  as in quad. constraints
3. if not separated, try interval gradient cut

**ENFORCE:**

4. if have linearization gap in  $x_{N_r}^\top A_r x_{N_r}$ , do branching
5. compute new quadratic underestimator for  $g_r$  that is tight at  $\hat{x}$   
(since Monday; at most 10 times per constraint)



Given function  $g(x)$  and box  $[x^L, x^U]$ .

Compute **interval gradient**

$$[d^L, d^U] = \nabla g([x^L, x^U]),$$

i.e.,  $\nabla g(x) \in [d^L, d^U]$  for all  $x \in [x^L, x^U]$ .

(TODO: better use slopes)



Given function  $g(x)$  and box  $[x^L, x^U]$ .

Compute **interval gradient**

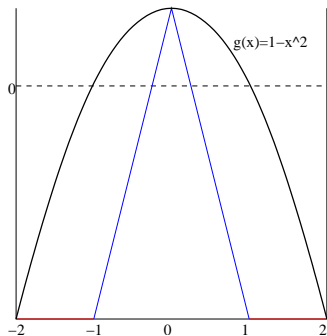
$$[d^L, d^U] = \nabla g([x^L, x^U]),$$

i.e.,  $\nabla g(x) \in [d^L, d^U]$  for all  $x \in [x^L, x^U]$ .

(TODO: better use slopes)

Let  $\hat{x} \in [x^L, x^U]$ . Then

$$g(\hat{x}) + \min_{d \in [d^L, d^U]} d^\top (x - \hat{x}) \leq g(x)$$



Given function  $g(x)$  and box  $[x^L, x^U]$ .

Compute **interval gradient**

$$[d^L, d^U] = \nabla g([x^L, x^U]),$$

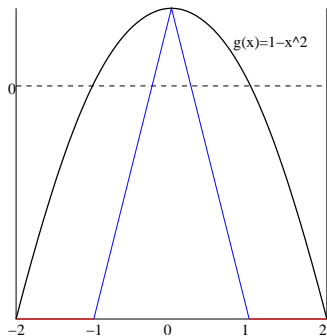
i.e.,  $\nabla g(x) \in [d^L, d^U]$  for all  $x \in [x^L, x^U]$ .

(TODO: better use slopes)

Let  $\hat{x} \in [x^L, x^U]$ . Then

$$g(\hat{x}) + \min_{d \in [d^L, d^U]} d^\top (x - \hat{x}) \leq g(x)$$

$$\Rightarrow g(\hat{x}) + \sum_{i: x_i \geq \hat{x}_i} d_i^L (x_i - \hat{x}_i) + \sum_{i: x_i < \hat{x}_i} d_i^U (x_i - \hat{x}_i) \leq g(x)$$



Given function  $g(x)$  and box  $[x^L, x^U]$ .

Compute **interval gradient**

$$[d^L, d^U] = \nabla g([x^L, x^U]),$$

i.e.,  $\nabla g(x) \in [d^L, d^U]$  for all  $x \in [x^L, x^U]$ .

(TODO: better use slopes)

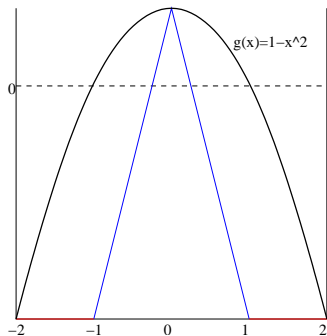
Let  $\hat{x} \in [x^L, x^U]$ . Then

$$g(\hat{x}) + \min_{d \in [d^L, d^U]} d^\top (x - \hat{x}) \leq g(x)$$

$$\Rightarrow g(\hat{x}) + \sum_{i: x_i \geq \hat{x}_i} d_i^L (x_i - \hat{x}_i) + \sum_{i: x_i < \hat{x}_i} d_i^U (x_i - \hat{x}_i) \leq g(x)$$

**Simple version** (implemented): move  $\hat{x}$  into vertex of box ( $\hat{x}_i \in \{x_i^L, x_i^U\}$ )

$$\Rightarrow g(\hat{x}) + \sum_{i: \hat{x}_i = x_i^L} d_i^L (x_i - \hat{x}_i) + \sum_{i: \hat{x}_i = x_i^U} d_i^U (x_i - \hat{x}_i) \leq g(x)$$



**PROPAGATE:**

- ▷ Consider a constraint

$$y \leq h(x) \quad \text{and let} \quad [h^L, h^U] := h([x^L, x^U])$$

(i.e.,  $h(x) \in [h^L, h^U]$  for all  $x \in [x^L, x^U]$ ).

**PROPAGATE:**

- ▷ Consider a constraint

$$y \leq h(x) \quad \text{and let} \quad [h^L, h^U] := h([x^L, x^U])$$

(i.e.,  $h(x) \in [h^L, h^U]$  for all  $x \in [x^L, x^U]$ ).

- ▷ If  $h^U < y^U$ , set

$$y^U := h^U$$

Proceed with other constraints depending on  $y$   
(follow a dependency graph).

**PROPAGATE:**

- ▷ Consider a constraint

$$y \leq h(x) \quad \text{and let} \quad [h^L, h^U] := h([x^L, x^U])$$

(i.e.,  $h(x) \in [h^L, h^U]$  for all  $x \in [x^L, x^U]$ ).

- ▷ If  $h^U < y^U$ , set

$$y^U := h^U$$

Proceed with other constraints depending on  $y$   
(follow a dependency graph).

- ▷ If  $y^U < h^L$ , then the node is **infeasible**.



- 1 Introduction
- 2 Constraint Handling
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule**
- 4 Upper Bounds
- 5 Preliminary Numerical Results
- 6 Last Minute Work...
- 7 Conclusions



Which variable to select for branching?

- ▷ if LP relaxation solution is **fractional**, choose **integer variable**  
(SCIP default: reliability branching on pseudo cost values)



Which variable to select for branching?

- ▷ if LP relaxation solution is **fractional**, choose **integer variable**  
(SCIP default: reliability branching on pseudo cost values)
- ▷ selection of **continuous variable**:  
look at [Belotti et.al.](#): Branching and bounds tightening techniques for non-convex MINLP (Pietro's talk from Monday)



Which variable to select for branching?

- ▷ if LP relaxation solution is **fractional**, choose **integer variable**  
(SCIP default: reliability branching on pseudo cost values)
- ▷ selection of **continuous variable**:  
look at **Belotti et.al.**: Branching and bounds tightening techniques for non-convex MINLP (Pietro's talk from Monday)
- ▷ search for pareto-optimal branching rule w.r.t.  
performance and ease of implementation  
⇒ **br-plain**



Which variable to select for branching?

- ▷ if LP relaxation solution is **fractional**, choose **integer variable** (SCIP default: reliability branching on pseudo cost values)
- ▷ selection of **continuous variable**:  
look at **Belotti et.al.**: Branching and bounds tightening techniques for non-convex MINLP (Pietro's talk from Monday)
- ▷ search for pareto-optimal branching rule w.r.t.  
performance and ease of implementation  
⇒ **br-plain**
- ▷ for constraint  $h_j(x) \leq u$  violated in  $\hat{x}$ , let  
 $\phi_j :=$  (scaled) **gap between  $h_j(\hat{x})$  and its best linear underestimator** in  $\hat{x}$



Which variable to select for branching?

- ▶ if LP relaxation solution is **fractional**, choose **integer variable** (SCIP default: reliability branching on pseudo cost values)
- ▶ selection of **continuous variable**:  
look at **Belotti et.al.**: Branching and bounds tightening techniques for non-convex MINLP (Pietro's talk from Monday)
- ▶ search for pareto-optimal branching rule w.r.t.  
performance and ease of implementation  
⇒ **br-plain**
- ▶ for constraint  $h_i(x) \leq u$  violated in  $\hat{x}$ , let  
 $\phi_i :=$  (scaled) **gap between  $h_i(\hat{x})$  and its best linear underestimator** in  $\hat{x}$
- ▶ select variable  $x_j$  with maximal **variable infeasibility**

$$0.1 \sum_{x_j \text{ in } h_i(x)} \phi_i + 1.3 \max_{x_j \text{ in } h_i(x)} \phi_i + 0.8 \min_{x_j \text{ in } h_i(x)} \phi_i$$



- 1 Introduction
- 2 Constraint Handling
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule
- 4 Upper Bounds**
- 5 Preliminary Numerical Results
- 6 Last Minute Work...
- 7 Conclusions



nothing fancy yet...

- ▶ solve (locally) MINLP with discrete variables fixed  $\Rightarrow$  NLP



nothing fancy yet...

- ▷ solve (locally) MINLP with discrete variables fixed  $\Rightarrow$  NLP
- ▷ get fixation of discrete variables from
  - ▶ integral solution of LP relaxation in nodes



nothing fancy yet...

- ▷ solve (locally) MINLP with discrete variables fixed  $\Rightarrow$  NLP
- ▷ get fixation of discrete variables from
  - ▶ integral solution of LP relaxation in nodes
  - ▶ integral solution found by one of SCIPs MIP heuristics



- 1 Introduction
- 2 Constraint Handling
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule
- 4 Upper Bounds
- 5 Preliminary Numerical Results**
- 6 Last Minute Work...
- 7 Conclusions



implemented as solver

## Solving **N**onlinear **I**nteger **P**rograms



implemented as solver

## Solving **N**onlinear **I**nteger **P**rograms

- ▷ GAMS interface
- ▷ function, gradient, hessian, interval evaluation by GAMS AD Library



implemented as solver

## Solving **N**onlinear **I**nteger **P**rograms

- ▷ GAMS interface
- ▷ function, gradient, hessian, interval evaluation by GAMS AD Library
- ▷ LP solver: CPLEX 10.0
- ▷ NLP solver: CONOPT 3



implemented as solver

## Solving Nonlinear Integer Programs

- ▷ GAMS interface
- ▷ function, gradient, hessian, interval evaluation by GAMS AD Library
- ▷ LP solver: CPLEX 10.0
- ▷ NLP solver: CONOPT 3

**Model instances:** 49 MINLPs from Pietro et.al.'s paper

- ▷ in average: 200 variables (105 discrete), 151 constraints, 92 nonlin. nonz.
- ▷ 19 convex, 13 nonconvex quadratic, 17 nonconvex nonquadratic



implemented as solver

## Solving Nonlinear Integer Programs

- ▷ GAMS interface
- ▷ function, gradient, hessian, interval evaluation by GAMS AD Library
- ▷ LP solver: CPLEX 10.0
- ▷ NLP solver: CONOPT 3

**Model instances:** 49 MINLPs from Pietro et.al.'s paper

- ▷ in average: 200 variables (105 discrete), 151 constraints, 92 nonlin. nonz.
- ▷ 19 convex, 13 nonconvex quadratic, 17 nonconvex nonquadratic

**Time limit:** 1 hour

**Gap tolerance:** 1%



instance	var	discr	nlz	time[s]	gap
du-opt	21	13	20	2.0	< 1%
du-opt5	21	12	20	1.1	< 1%
fo7	115	42	14	172.7	< 1%
m6	87	30	12	4.1	< 1%
no7_ar2_1	113	42	14	34.3	< 1%
no7_ar3_1	113	42	14	449.5	< 1%
no7_ar4_1	113	42	14	361.1	< 1%
o7_2	115	42	14	2946.0	< 1%
stockcycle	481	432	48	3600.0	4%
tls5	162	136	50	3600.0	63%
tls6	216	179	72	3600.0	150%
ibell3a	123	60	60	0.6	< 1%
ibienst1	506	28	28	60.3	< 1%
classical_40_0	121	40	40	17.8	< 1%
classical_40_1	121	40	40	13.5	< 1%
robust_30_0	124	31	61	3600.0	4.3%
robust_30_1	124	31	61	3600.0	4.4%
shortfall_30_0	125	31	62	20.4	< 1%
shortfall_30_1	125	31	62	514.1	< 1%



instance	var	discr	nlz	time[s]	gap
lop97icx	987	899	704	3600.0	38%
nous1	51	2	122	3600.0	65%
nous2	51	2	122	124.1	< 1%
nvs19	9	8	72	602.2	< 1%
nvs23	10	9	90	3600.0	99%
space25	894	750	111	3600.0	$\infty$
space25a	384	240	111	3600.0	737%
tln5	36	35	50	35.4	< 1%
tln6	49	48	72	3600.0	44%
tln7	64	63	98	3600.0	107%
tln12	169	168	288	3600.0	523%
imisc07	261	259	259	81.6	< 1%
iran8x32	513	256	256	101.2	< 1%



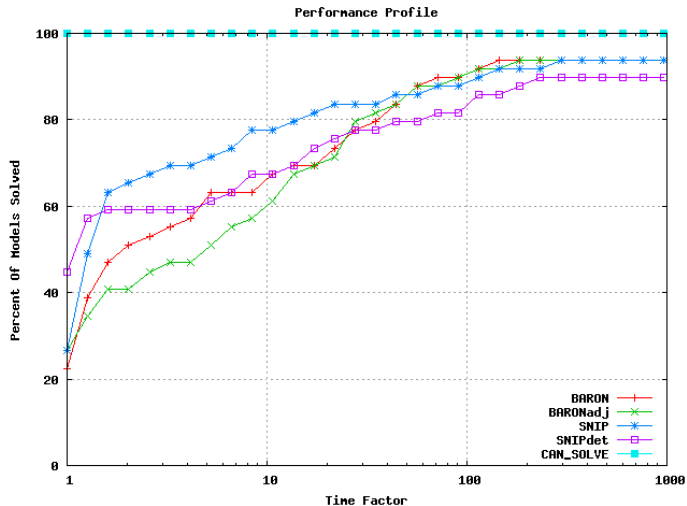
instance	var	discr	n nz	time[s]	gap
cecil_13	841	162	360	863.2	< 1%
csched1	77	63	8	0.8	< 1%
csched2	401	308	58	3600.0	80%
eniplac	142	24	48	206.9	< 1%
enpro48pb	154	92	29	2.3	< 1%
enpro56pb	128	73	24	24.2	< 1%
ex1233	53	12	28	311.1	< 1%
ex1243	69	16	36	20.8	< 1%
ex1244	96	23	52	262.6	< 1%
ex1252	40	15	36	395.0	< 1%
synheat	57	12	28	3600.0	12%
c-sched-4-7	234	140	57	3600.0	100%
clay0203h	91	18	72	15.2	< 1%
clay0204h	165	32	96	28.2	< 1%
multistage	186	18	55	3600.0	99%
par72	569	56	406	3600.0	$\infty$
synheatmod	54	12	28	206.1	< 1%



- ▷ BARONadj = BARON with reduced probing (PEnd=10, PDo=100)
- ▷ SNIPdet = SNIP without quadratic under/overestimators

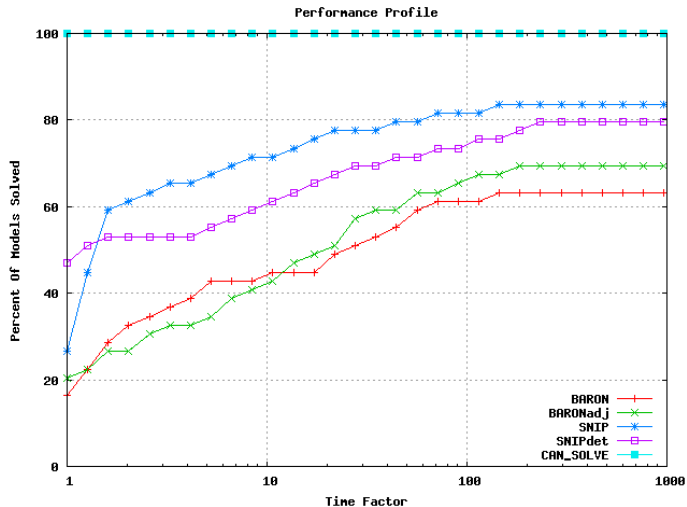


Metric: "Solved" = found a feasible solution



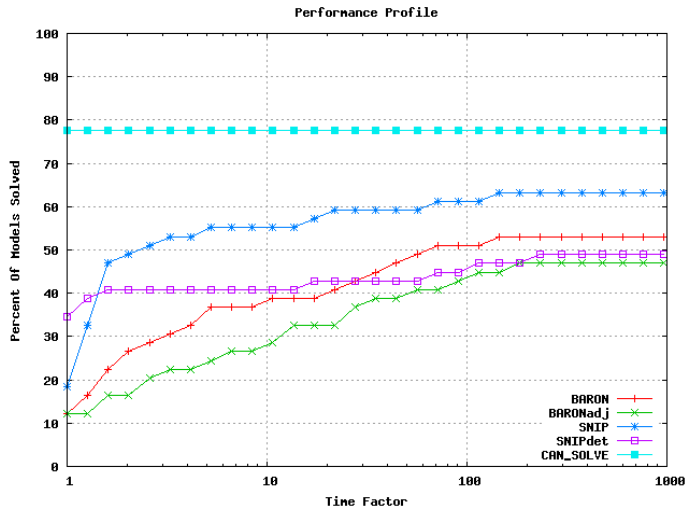
- ▷ BARONadj = BARON with reduced probing (PEnd=10, PDo=100)
- ▷ SNIPdet = SNIP without quadratic under/overestimators

Metric: “Solved” = found **best** feasible solution (among all solvers)



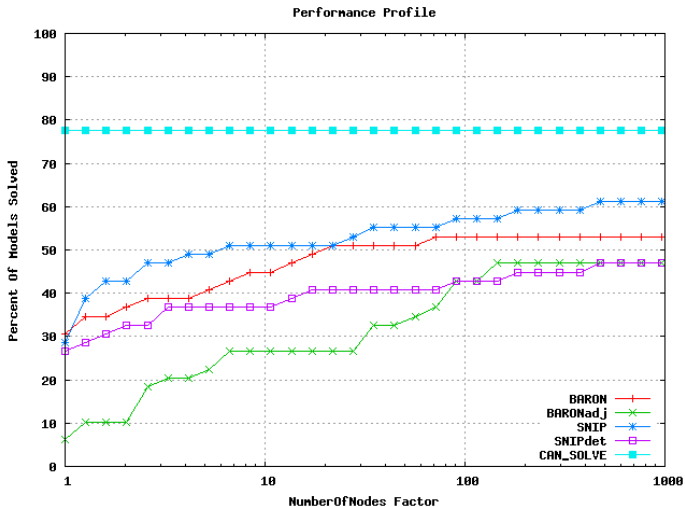
- ▷ BARONadj = BARON with reduced probing (PEnd=10, PDo=100)
- ▷ SNIPdet = SNIP without quadratic under/overestimators

Metric: "Solved" = gap < 1%



- ▷ BARONadj = BARON with reduced probing (PEnd=10, PDo=100)
- ▷ SNIPdet = SNIP without quadratic under/overestimators

Metric: "Solved" = gap < 1%; number of **B&B-nodes** (instead of time)



- ▷ BARONadj = BARON with reduced probing (PEnd=10, PDo=100)
- ▷ SNIPdet = SNIP without quadratic under/overestimators



- 1 Introduction
- 2 Constraint Handling
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule
- 4 Upper Bounds
- 5 Preliminary Numerical Results
- 6 Last Minute Work...**
- 7 Conclusions



- ▷ recall **interval arithmetic** based boundtightening

$$y \leq h(x), \quad h(x) \in [h^L, h^U] \quad \Rightarrow \quad y^U \leq h^U$$

- ▷ does not infer bounds on nonlinear variables  $x$



- ▷ recall **interval arithmetic** based boundtightening

$$y \leq h(x), \quad h(x) \in [h^L, h^U] \quad \Rightarrow \quad y^U \leq h^U$$

- ▷ does not infer bounds on nonlinear variables  $x$
- ▷ could walk through expression tree to infer variables bounds from constraint bounds



Represent **algebraic structure** of problem in a directed acyclic graph (DAG):

- ▷ nodes: variables, operations, constraints
- ▷ arcs: flow of computation



Represent algebraic structure of problem in a directed acyclic graph (DAG):

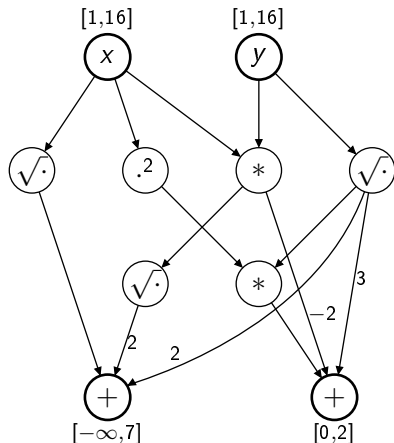
- ▷ nodes: variables, operations, constraints
- ▷ arcs: flow of computation

Example:

$$\sqrt{x} + 2\sqrt{xy} + 2\sqrt{y} \in [-\infty, 7]$$

$$x^2\sqrt{y} - 2xy + 3\sqrt{y} \in [0, 2]$$

$$x, y \in [1, 16]$$





Represent **algebraic structure** of problem in a directed acyclic graph (DAG):

- ▶ nodes: variables, operations, constraints
- ▶ arcs: flow of computation

**Example:**

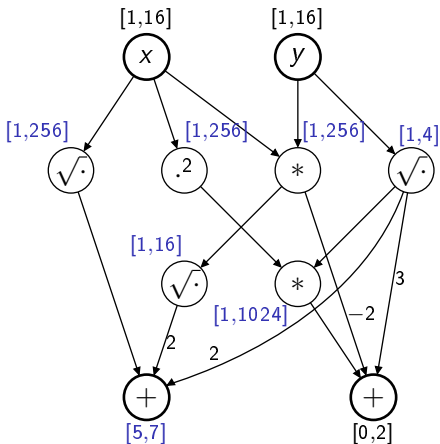
$$\sqrt{x} + 2\sqrt{xy} + 2\sqrt{y} \in [-\infty, 7]$$

$$x^2\sqrt{y} - 2xy + 3\sqrt{y} \in [0, 2]$$

$$x, y \in [1, 16]$$

**Forward propagation:**

- ▶ compute bounds on intermediate nodes (top-down)





Represent algebraic structure of problem in a directed acyclic graph (DAG):

- ▶ nodes: variables, operations, constraints
- ▶ arcs: flow of computation

Example:

$$\sqrt{x} + 2\sqrt{xy} + 2\sqrt{y} \in [-\infty, 7]$$

$$x^2\sqrt{y} - 2xy + 3\sqrt{y} \in [0, 2]$$

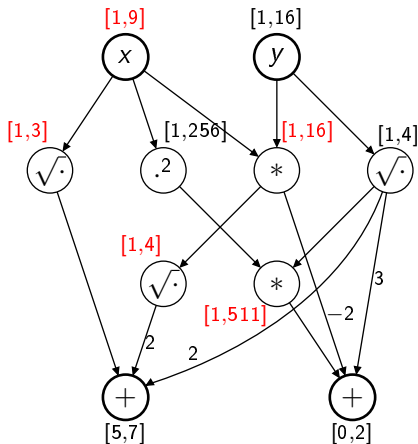
$$x, y \in [1, 16]$$

Forward propagation:

- ▶ compute bounds on intermediate nodes (top-down)

Backward propagation:

- ▶ reduce bounds using reverse operations (bottom-up)





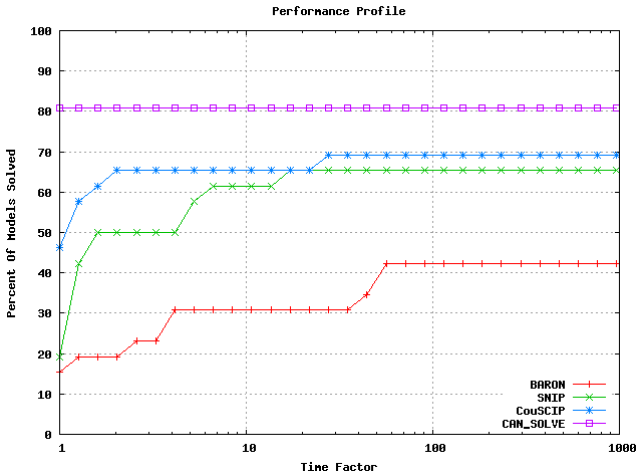
- ▷ Couenne [Belotti et.al.] has “feasibility-based bound tightening” (c.f. poster)



- ▷ Couenne [Belotti et.al.] has “feasibility-based bound tightening” (c.f. poster)
- ▷ “steal their ~~plan~~ code”: substitutes LaGO’s bound tightening for nonquadratic constraints by Couenne



- ▷ Couenne [Belotti et.al.] has “feasibility-based bound tightening” (c.f. poster)
- ▷ “steal their plan code”: substitutes LaGO’s bound tightening for nonquadratic constraints by Couenne



(26 nonquad. MINLPs; “solved” = gap < 1%)



- 1 Introduction
- 2 Constraint Handling
  - Quadratic Constraints (MIQQP)
  - Nonlinear Constraints (MINLP) - LaGO
- 3 Branching Rule
- 4 Upper Bounds
- 5 Preliminary Numerical Results
- 6 Last Minute Work...
- 7 Conclusions



Among many things...

▷ better handling of **quadratic** constraints

[Saxena, Bonami, Lee '08], [Tawarmalani, Richard, Chung '08], [Jach, Michaels, Weismantel '08], [Liberti, Pantelides '05]



Among many things...

- ▷ better handling of **quadratic** constraints

[Saxena, Bonami, Lee '08], [Tawarmalani, Richard, Chung '08], [Jach, Michaels, Weismantel '08], [Liberti, Pantelides '05]

- ▷ **SOC** [Ben-Tal, Nemirovski '01], [Vielma, Ahmed, Nemhauser '05], [Drewes '09]



Among many things...

- ▷ better handling of **quadratic** constraints  
[Saxena, Bonami, Lee '08], [Tawarmalani, Richard, Chung '08], [Jach, Michaels, Weismantel '08], [Liberti, Pantelides '05]
- ▷ **SOC** [Ben-Tal, Nemirovski '01], [Vielma, Ahmed, Nemhauser '05], [Drewes '09]
- ▷ **preprocessing**



Among many things...

- ▷ better handling of **quadratic** constraints  
[Saxena, Bonami, Lee '08], [Tawarmalani, Richard, Chung '08], [Jach, Michaels, Weismantel '08], [Liberti, Pantelides '05]
- ▷ **SOC** [Ben-Tal, Nemirovski '01], [Vielma, Ahmed, Nemhauser '05], [Drewes '09]
- ▷ **preprocessing**

Thank you!