Global Optimization of Mixed-Integer Nonlinear Programs with SCIP 8

Ksenia Bestuzheva, Antonia Chmiela, Benjamin Müller, Felipe Serrano, Stefan Vigerske, Fabian Wegscheider

Kyushu University
January, 20th, 2023

GAMS Software GmbH and Zuse Institute Berlin

TOC

Introduction

What is SCIP?

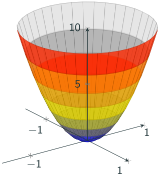
SCIP (Solving Constraint Integer Programs) ...

- provides a full-scale MIP and MINLP solver,
- incorporates
  - MIP features (cutting planes, LP relaxation), and
  - MINLP features (spatial branch-and-bound, NLP relaxation)
  - CP features (domain propagation),
  - SAT-solving features (conflict analysis, restarts),
- is a branch-cut-and-price framework,
- has a modular structure via plugins,
- is open-source (since November 2022),
- and is available under https://www.scipopt.org.

Mixed-Integer Nonlinear Programming

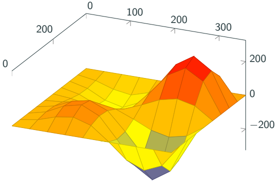
$$\begin{aligned} \min\ & c^\mathsf{T} x \\ \text{s.t. } & g_k(x) \leq 0 && \forall k \in [m] \\ & x_i \in \mathbb{Z} && \forall i \in \mathcal{I} \subseteq [n] \\ & x_i \in [\ell_i, u_i] && \forall i \in [n] \end{aligned}$$

The functions $g_k : [\ell, u] \to \mathbb{R}$ can be



convex

or



nonconvex

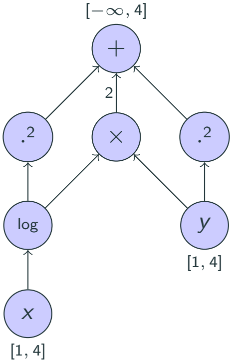
and are given in algebraic form.

Expression Trees

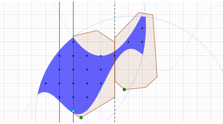
The algebraic structure of nonlinear constraints is stored in a directed acyclic graph:

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

$$\log(x)^2 + 2\log(x)y + y^2 \in [-\infty, 4]$$
$$x, y \in [1, 4]$$



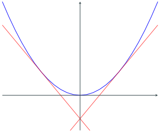
Spatial Branch and Bound



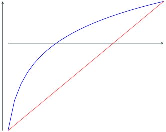
SCIP solves MINLPs by spatial Branch & Bound.

LP relaxation via convexification and linearization:

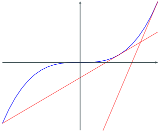
convex functions



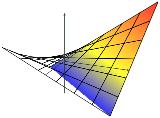
concave functions



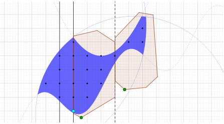
$x^k \quad (k \in 2\mathbb{Z}+1)$



$x \cdot y$

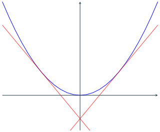
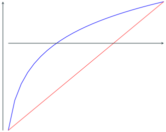
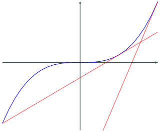
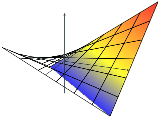


Spatial Branch and Bound

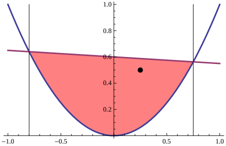


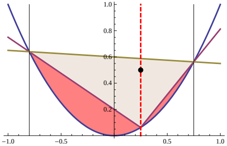
SCIP solves MINLPs by spatial Branch & Bound.

LP relaxation via convexification and linearization:

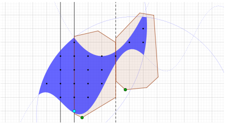
convex functions	concave functions	$x^k \quad (k \in 2\mathbb{Z}+1)$	$x \cdot y$
			

Branching on variables in violated nonconvex constraints:





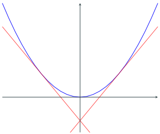
Spatial Branch and Bound



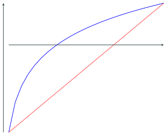
SCIP solves MINLPs by spatial Branch & Bound.

LP relaxation via convexification and linearization:

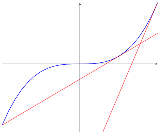
convex functions



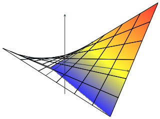
concave functions



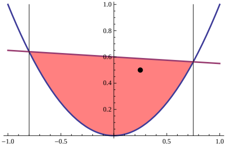
$x^k \quad (k \in 2\mathbb{Z}+1)$

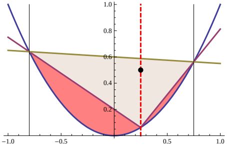


$x \cdot y$



Branching on variables in violated nonconvex constraints:





. . . and bound tightening (FBBT, OBBT), primal heuristics (e.g., sub-NLP/MIP/MINLP), other special techniques

SCIP 7 (and before)

Reformulation in Presolve

Goal: Reformulate constraints such that only elementary cases (convex, concave, odd power, quadratic) remain by introducing new variables and new constraints.

Consider

$$\begin{aligned} &\min z \\ &\text{s.t. } \exp(\ln(1000) + 1 + x\,y) \le z \\ &\qquad x^2 + y^2 \le 2 \end{aligned}$$

Reformulation in Presolve

Goal: Reformulate constraints such that only elementary cases (convex, concave, odd power, quadratic) remain by introducing new variables and new constraints.

Consider

$$\begin{aligned} &\min z \\ &\text{s.t. } \exp(\ln(1000) + 1 + x\,y) \leq z \\ &\qquad x^2 + y^2 \leq 2 \end{aligned}$$

Reformulation takes apart $\exp(\ln(1000) + 1 + x\,y)$, thus SCIP actually solves the extended formulation

$$\begin{aligned} &\min z \\ &\text{s.t. } \exp(w) \leq z \\ &\qquad \ln(1000) + 1 + x\,y = w \\ &\qquad x^2 + y^2 \leq 2 \end{aligned}$$

Issue with explicit reformulation

SCIP solves reformulated problem fine:

```
SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.08
Solving Nodes      : 5
Primal Bound       : +9.99999656552062e+02 (3 solutions)
Dual Bound         : +9.99999656552062e+02
Gap                : 0.00 %
```

Issue with explicit reformulation

SCIP solves reformulated problem fine:

```
SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.08
Solving Nodes      : 5
Primal Bound       : +9.99999656552062e+02 (3 solutions)
Dual Bound         : +9.99999656552062e+02
Gap                : 0.00 %
```

Solution $(x, y, z, w) = (-1.000574549, 0.999425451, 999.999656552, 6.907754936)$ looks ok:

	Violation
$\min z$	
s.t. $\exp(w) \le z$	$0.4659 \cdot 10^{-6} \le \texttt{feastol}$ ✓
$\ln(1000) + 1 + x\,y = w$	$0.6731 \cdot 10^{-6} \le \texttt{feastol}$ ✓
$x^2 + y^2 \le 2$	$0.6602 \cdot 10^{-6} \le \texttt{feastol}$ ✓

Issue with explicit reformulation

SCIP solves reformulated problem ~~fine~~:

```
SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.08
Solving Nodes      : 5
Primal Bound       : +9.99999656552062e+02 (3 solutions)
Dual Bound         : +9.99999656552062e+02
Gap                : 0.00 %
```

Solution $(x, y, z, w) = (-1.000574549, 0.999425451, 999.999656552, 6.907754936)$ looks ok:

	Violation
$\min z$	
s.t. $\exp(w) \leq z$	$0.4659 \cdot 10^{-6} \leq \texttt{feastol}$ ✓
$\ln(1000) + 1 + x\,y = w$	$0.6731 \cdot 10^{-6} \leq \texttt{feastol}$ ✓
$x^2 + y^2 \leq 2$	$0.6602 \cdot 10^{-6} \leq \texttt{feastol}$ ✓

However, original $\exp(\ln(1000) + 1 + x\,y) \leq z$ has too large violation:

```
[nonlinear] <e1>: exp((7.9077552789821368151 +1 (<x> * <y>)))-1<z>[C]  <= 0;
violation: right hand side is violated by 0.000673453314561812
best solution is not feasible in original problem
```

Problem with classic approach

$\Rightarrow$ Explicit reformulation of constraints ...

- ... loses the connection to the original problem.
- ... loses distinction between original and auxiliary variables.
  Thus, we may branch on auxiliary variables.
- ... prevents simultaneous exploitation of overlapping structures.

SCIP 8: updated framework

Main Ideas

Avoid explicit split-up of constraints.

- introduce extended formulation as annotation to the original formulation
- use extended formulation for relaxation
- use original formulation for feasibility checking
- to resolve infeasibility in original constraints, tighten relaxation of extended formulation

Main Ideas

Avoid explicit split-up of constraints.

- introduce extended formulation as annotation to the original formulation
- use extended formulation for relaxation
- use original formulation for feasibility checking
- to resolve infeasibility in original constraints, tighten relaxation of extended formulation

Everything nonlinear is an expression.

- represent all nonlinear constraints in one expression graph (DAG)
- all algorithms (check, separation, propagation, etc.) work on the same expression graph, no more specialized nonlinear constraints

Main Ideas

Avoid explicit split-up of constraints.

- introduce extended formulation as annotation to the original formulation
- use extended formulation for relaxation
- use original formulation for feasibility checking
- to resolve infeasibility in original constraints, tighten relaxation of extended formulation

Everything nonlinear is an expression.

- represent all nonlinear constraints in one expression graph (DAG)
- all algorithms (check, separation, propagation, etc.) work on the same expression graph, no more specialized nonlinear constraints
- separate expression operators ($+$, $\times$) and high-level structures (quadratic, semi-continuous, second order cone, etc.)

Expression Handlers

Each operator type ($+$, $\times$, pow, etc.) is implemented by an **expression handler**, which can provide a number of callbacks:

- evaluate and differentiate expression w.r.t. operands
- interval evaluation and tighten bounds on operands
- provide linear under- and over-estimators
- inform about curvature, monotonicity, integrality
- simplify, compare, print, parse, hash, copy, etc.

Expression handlers are like other **SCIP plugins**.
New ones can be added by users (YOU!).

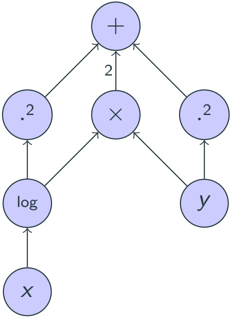
Available handler: abs, cos, entropy, exp, log, pow, product, signpow, sin, sum, value, var

Example: Extended Formulation (exprhdlr only)

Constraint:

$$\log(x)^2 + 2\log(x)y + y^2 \le 4$$

This formulation is used to check feasibility and presolve.



Example: Extended Formulation (exprhdlr only)

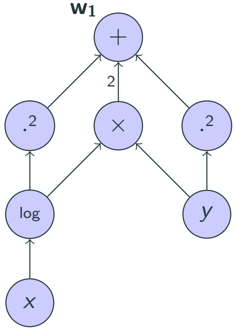
Constraint:

$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to check feasibility and presolve.

Extended Formulation:

$$w_1 \leq 4$$
$$\log(x)^2 + 2\log(x)y + y^2 = w_1$$



Example: Extended Formulation (exprhdlr only)

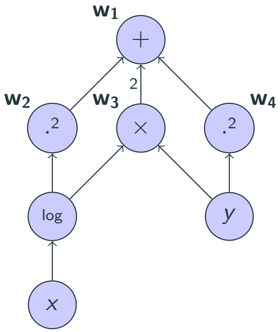
Constraint:

$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to check feasibility and presolve.

Extended Formulation:

$$\begin{aligned} w_1 &\leq 4 \\ w_2 + 2w_3 + w_4 &= w_1 \\ \log(x)^2 &= w_2 \\ \log(x)y &= w_3 \\ y^2 &= w_4 \end{aligned}$$



Example: Extended Formulation (exprhdlr only)

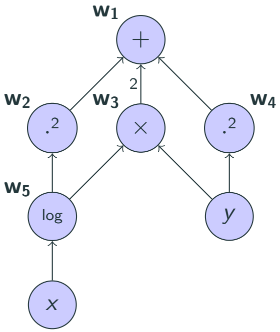
Constraint:

$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to check feasibility and presolve.

Extended Formulation:

$$\begin{aligned} w_1 &\leq 4 \\ w_2 + 2w_3 + w_4 &= w_1 \\ w_5^2 &= w_2 \\ w_5 y &= w_3 \\ y^2 &= w_4 \\ \log(x) &= w_5 \end{aligned}$$



Used to construct LP relaxation.

Nonlinearity Handlers

But $\log(x)^2 + 2\log(x)y + y^2 \leq 4$ is convex and quadratic in $(\log(x), y)$.

Nonlinearity Handlers

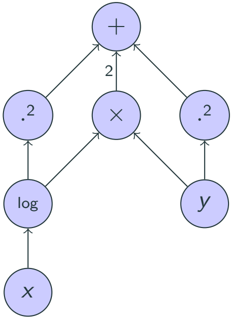
But $\log(x)^2 + 2\log(x)y + y^2 \leq 4$ is convex and quadratic in $(\log(x), y)$.

To explore structure, we now have **Nonlinearity Handler**:

- Adds additional **separation** and/or **propagation** algorithms for structures that can be identified in the expression graph.
- Attached to nodes in expression graph, but does not define expressions nor constraints.
- Examples: quadratics, convex and concave, second order cone, ...
- Several nlhdlrs can be attached to a node in the expression graph.

Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \leq 4$

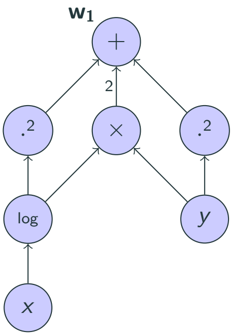


Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \leq 4$

$w_1 \leq 4$

1. Annotate root with auxiliary variable w_1.

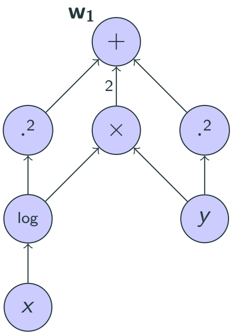


Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \leq 4$

$w_1 \leq 4$

1. Annotate root with auxiliary variable w_1.
2. Run `detect` of all nlhdlrs on $+$ node.



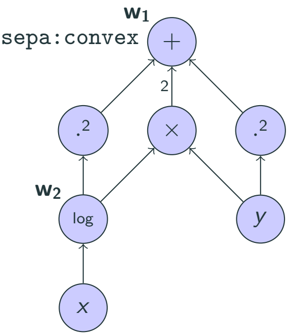
Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \leq 4$

1. Annotate root with auxiliary variable w_1.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_convex`
     - detects a convex quadratic structure,
     - signals that it can compute underestimators,
     - but requests an auxiliary variable w_2 for log node.

$$w_1 \leq 4$$

$$w_2^2 + 2w_2 y + y^2 \leq w_1$$



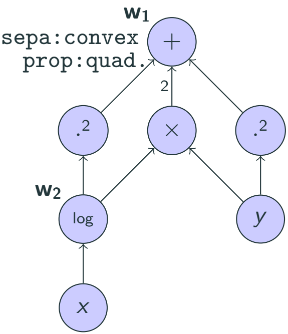
Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \leq 4$

1. Annotate root with auxiliary variable w_1.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_convex`
     - detects a convex quadratic structure,
     - signals that it can compute underestimators,
     - but requests an auxiliary variable w_2 for log node.
   - `nlhdlr_quadratic`
     - also detects a quadratic structure,
     - signals that it can do domain propagation, and
     - notifies that it will use bounds of nodes log and y.

$$w_1 \leq 4$$

$$w_2^2 + 2w_2 y + y^2 \leq w_1$$



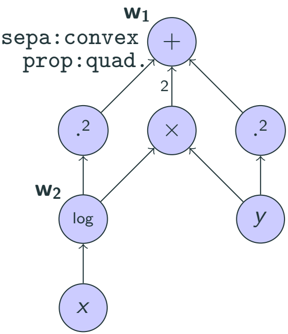
Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \leq 4$

1. Annotate root with auxiliary variable w_1.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_convex`
     - detects a convex quadratic structure,
     - signals that it can compute underestimators,
     - but requests an auxiliary variable w_2 for log node.
   - `nlhdlr_quadratic`
     - also detects a quadratic structure,
     - signals that it can do domain propagation, and
     - notifies that it will use bounds of nodes log and y.
3. Run `detect` of all nlhdlrs on log node.

$$w_1 \leq 4$$

$$w_2^2 + 2w_2y + y^2 \leq w_1$$

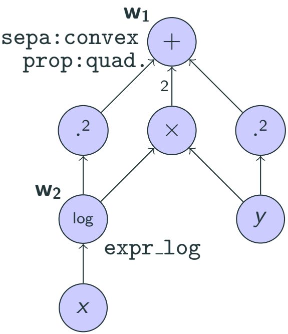


Example: Extended Formulation (with nlhdlrs)

Constraint: $\log(x)^2 + 2\log(x)y + y^2 \le 4$

1. Annotate root with auxiliary variable w_1.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_convex`
     - detects a convex quadratic structure,
     - signals that it can compute underestimators,
     - but requests an auxiliary variable w_2 for log node.
   - `nlhdlr_quadratic`
     - also detects a quadratic structure,
     - signals that it can do domain propagation, and
     - notifies that it will use bounds of nodes log and y.
3. Run `detect` of all nlhdlrs on log node.
   - No specialized nlhdlr signals success.
     The expression handler will be used for both under/overestimation and propagation.

$$w_1 \le 4$$
$$w_2^2 + 2w_2 y + y^2 \le w_1$$
$$\log(x) = w_2$$



MINLP features in SCIP (teasers only)

SCIP features particular for nonlinear structures

$+$ = new or improved in SCIP 8

Presolve

- simplify expressions

+ identify common subexpressions

Presolve

- simplify expressions

+ identify common subexpressions

+ linearize products of binary variables

Presolve

- simplify expressions

\+ identify common subexpressions

\+ linearize products of binary variables

- try to restrict variables appearing in one constraint only to its bounds,
  e.g., $xy + yz \leq u \to y \in \{\ell_y, u_y\}$ [Hansen et.al., 1993]

Presolve

- simplify expressions

+ identify common subexpressions

+ linearize products of binary variables

- try to restrict variables appearing in one constraint only to its bounds,
  e.g., $xy + yz \leq u \rightarrow y \in \{\ell_y, u_y\}$ [Hansen et.al., 1993]

- if QP, add KKT as redundant constraints

Presolve

- simplify expressions

\+ identify common subexpressions

\+ linearize products of binary variables

- try to restrict variables appearing in one constraint only to its bounds, e.g., $xy + yz \leq u \rightarrow y \in \{\ell_y, u_y\}$ [Hansen et.al., 1993]
- if QP, add KKT as redundant constraints

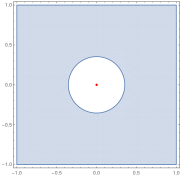
\+ symmetry detection using expression graph [Liberti 2010, Wegscheider 2019]

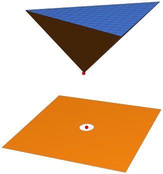
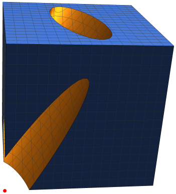
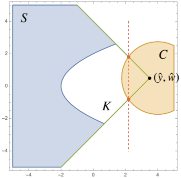
Quadratics

Nonlinear handler for quadratic subexpressions:

- provides domain propagation (variable bound tightening)

\+ intersection cuts for nonconvex quadratics [Chmiela, Muñoz, Serrano 2021]



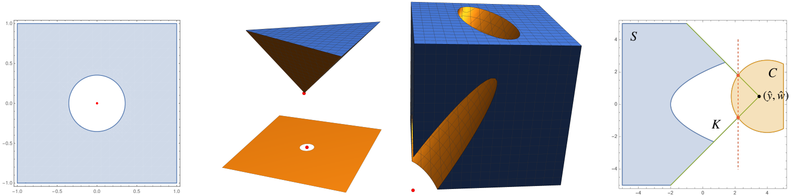
\+ also as separator for implied quadratics det(2×2 minors of X) $= 0$, $X = xx^{\top}$

Quadratics

Nonlinear handler for quadratic subexpressions:

- provides domain propagation (variable bound tightening)

\+ intersection cuts for nonconvex quadratics [Chmiela, Muñoz, Serrano 2021]



\+ also as separator for implied quadratics $\det(2 \times 2 \text{ minors of } X) = 0$, $X = xx^{\mathsf{T}}$

Separator for implied PSD constraint ($X \succeq xx^{\mathsf{T}}$):

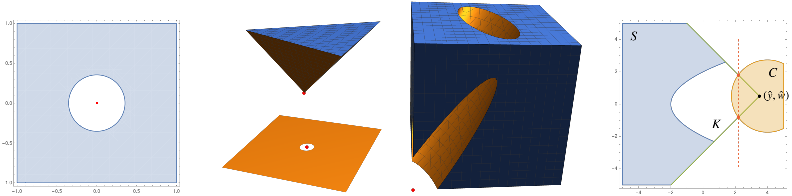
\+ SDP-cuts for 2×2 principal minors of $X - xx^{\mathsf{T}} \succeq 0$

Quadratics

Nonlinear handler for quadratic subexpressions:

- provides domain propagation (variable bound tightening)

\+ intersection cuts for nonconvex quadratics [Chmiela, Muñoz, Serrano 2021]



\+ also as separator for implied quadratics det(2×2 minors of X) $= 0$, $X = xx^{\mathsf{T}}$

Separator for implied PSD constraint ($X \succeq xx^{\mathsf{T}}$):

\+ SDP-cuts for 2×2 principal minors of $X - xx^{\mathsf{T}} \succeq 0$

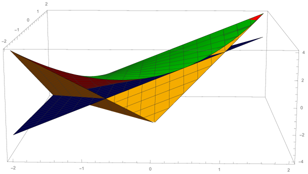
Separator for edge-concave quadratics:

- aggregate quadratic constraints to be edge-concave
- separate facets from vertex-polyhedral convex hull [Misener, Floudas 2012]

Bilinear

Nonlinear handler for bilinear expressions:

- convexify and domain propagation for xy w.r.t. additional inequalities on x, y, e.g., $x \leq y$



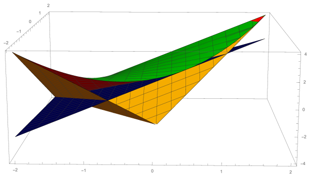
- inequalities found by 2D-projection of LP relaxation ($\approx$ OBBT)

[Linderoth 2004, Hijazi 2015, Locatelli 2016, Müller, Serrano, Gleixner 2020]

Bilinear

Nonlinear handler for bilinear expressions:

- convexify and domain propagation for xy w.r.t. additional inequalities on x, y, e.g., $x \leq y$



- inequalities found by 2D-projection of LP relaxation ($\approx$ OBBT)

[Linderoth 2004, Hijazi 2015, Locatelli 2016, Müller, Serrano, Gleixner 2020]

Reformulation Linearization Technique for bilinear products:

+ cuts from multiplication of LP rows and bounds
+ also for implicit products in mixed-binary linear problems

[Adams, Sherali 1986, Achterberg, Bestuzheva, Gleixner 2022]

Nonlinear handler for Second-Order Cones:

+ detect SOC constraints from quadratics and some Euclidean norms

Nonlinear handler for Second-Order Cones:

+ detect SOC constraints from quadratics and some Euclidean norms
- separate using disaggregated formulation

[Vielma, et.al. 2016]

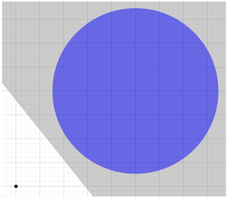
$\Downarrow$

Convexity and Concavity

Nonlinear handler for convex and concave expressions:

+ find convex/concave subexpressions using composition rules
  - gradient cuts on convex functions
  - facets of convex hull on concave function
+ prefer extended formulations for convex case

[Tawarmalani Sahinidis, 2005]

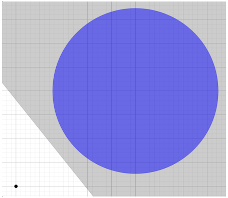


Convexity and Concavity

Nonlinear handler for convex and concave expressions:

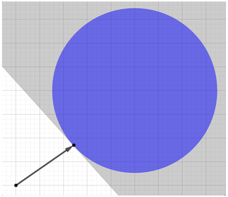
+ find convex/concave subexpressions using composition rules
  - gradient cuts on convex functions
  - facets of convex hull on concave function
+ prefer extended formulations for convex case

[Tawarmalani Sahinidis, 2005]



Separator for supporting hyperplanes:

- separators to linearize at boundary of convex NLP relaxation
- projection of given point, or linesearch to interior point



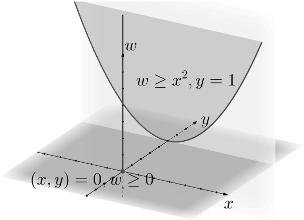
[Veinott 1967, Kronqvist, Lundell, Westerlund 2016, Serrano, Schwarz, Gleixner 2020]

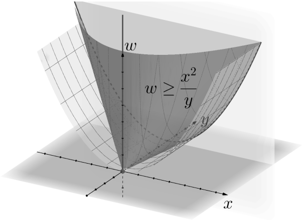
Perspective Strengthening

Nonlinear handler for perspective strengthening:

+ detect expressions in semi-continuous variables ($\ell_x y \leq x \leq u_x y$, $y \in \{0, 1\}$)
+ strengthen under/overestimators for such expressions, e.g.,

$$f(\hat{x}) + \nabla f(\hat{x})(x - \hat{x}) \leq w \quad \Rightarrow \quad f(\hat{x})y + \nabla f(\hat{x})(x - \hat{x}y) \leq w$$





[Frangioni, Gentile 2006, Bestuzheva, Gleixner, Vigerske 2021]

Quotients

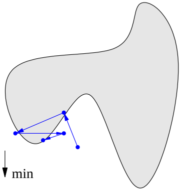
Nonlinear handler for quotients:

- to avoid ambiguity, there is no expression type for quotients

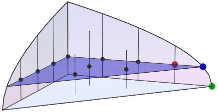
+ detect $\frac{ax+b}{cy+d}$ in nonlinear handler
+ provide bound tightening
+ provide linear under/overestimates [Zamora, Grossmann 1998]

Primal Heuristics

- fix all integer vars, solve NLP to local optimality



- multistart with constraint consensus for NLPs [Smith, Chinneck, Aitken 2013]
- NLP-diving
- solve sequence of regularized NLP reformulations of MINLP ("MPEC")
- fix nonlinear vars, solve remaining sub-MIP [Berthold, Gleixner 2014]

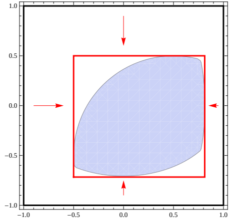


- other large neighborhood search heuristics, solving sub-MINLPs

[Berthold, Heinz, Pfetsch, Vigerske 2011]

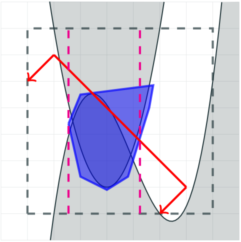
NLP solving via Ipopt and CppAD

- **Feasibility-Based Bound Tightening** (FBBT): as in constraint programming



- **Optimization-Based Bound Tightening** (OBBT): optimize variable over LP

[Gleixner, Berthold, Müller, Weltge 2017]



- OBBT over NLP relaxation

Interfaces

Readers for MINLP included:

- MPS, LP, PIP, AMPL NL, OSiL, ZIMPL

Interfaces:

- C, Java
- Python
- Julia, Matlab
- AMPL, GAMS
- ...

Benchmark

Solvers

Global MINLP solvers included in GAMS 41.2.0 (November 2022) and still maintained:

- **BARON** 22.9.30: commercial solver by The Optimization Firm (Nick Sahinidis)
  chooses between several LP/MIP/NLP subsolvers (CONOPT, CPLEX, ...)
- **Lindo API** 14.0.5099.162: commercial solver by Lindo Systems, Inc.
  uses CONOPT and MOSEK as subsolvers
- **Octeract** 4.5.1: commercial solver by Octeract, Ltd.
  uses CPLEX as LP/MIP/QP/QCP solver, Ipopt as NLP solver
- **SCIP** 8.0.2: open-source academic solver
  uses CPLEX as LP solver, Ipopt (with MA27) as NLP solver

Benchmark Setting

Test set:

- selected 200 instances from MINLPLib:
  - all solvers can handle (no sine, cosine, signpower)
  - solvable by at least one solver, but not trivial for all
  - varying degree of integrality
  - varying degree of nonlinearity
  - avoid too many instances with similar name
- 4 additional permutations of variables/equations $\Rightarrow$ **1000 instances**

Benchmark Setting

Test set:

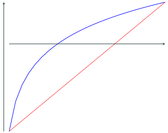
- selected 200 instances from MINLPLib:
  - all solvers can handle (no sine, cosine, signpower)
  - solvable by at least one solver, but not trivial for all
  - varying degree of integrality
  - varying degree of nonlinearity
  - avoid too many instances with similar name
- 4 additional permutations of variables/equations $\Rightarrow$ **1000 instances**

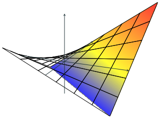
Settings:

- relative gap tolerance 10^{-4}, absolute gap tolerance 10^{-6}
- feasibility tolerance 10^{-6}
- bound unbounded variables by 10^{12}
- 2 hours time limit
- 1 thread/process
- 50 GB RAM

Why "bound unbounded variables by 10^{12}" ???

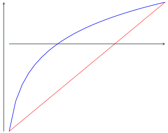
- convex underestimators typically require variable bounds
- without bounds, the relaxation may be weak or even unbounded

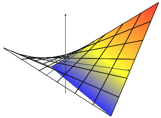




Why "bound unbounded variables by 10^{12}" ???

- convex underestimators typically require variable bounds
- without bounds, the relaxation may be weak or even unbounded



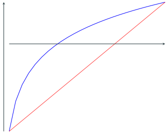


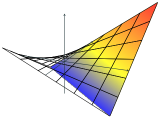
Solvers handle unbounded variables in different ways by default:

- SCIP skips relaxations and hopes that branching will help $\Rightarrow$ may not terminate
- BARON sets missing bounds of vars in nonconvex terms to $\approx \pm 10^{10}$; does not claim global optimality anymore
- Lindo API reduces bounds of all vars in nonconvex terms to $\pm 10^{10}$; may still claim optimality
- Octeract sets missing bounds of all variables to $\pm 10^{7}$ (!); may still claim optimality

Why “bound unbounded variables by 10^{12}” ???

- convex underestimators typically require variable bounds
- without bounds, the relaxation may be weak or even unbounded





Solvers handle unbounded variables in different ways by default:

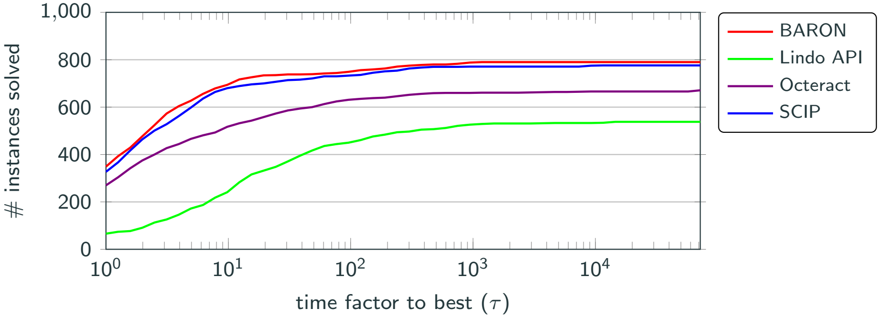
- SCIP skips relaxations and hopes that branching will help $\Rightarrow$ may not terminate
- BARON sets missing bounds of vars in nonconvex terms to $\approx \pm 10^{10}$; does not claim global optimality anymore
- Lindo API reduces bounds of all vars in nonconvex terms to $\pm 10^{10}$; may still claim optimality
- Octeract sets missing bounds of <u>all</u> variables to $\pm 10^{7}$ (!); may still claim optimality

$\Rightarrow$ **Default settings may mean that each solver solves a different subproblem of the actual problem and may report a lower bound of the subproblem only.**

Via parameters, we can enforce a more similar treatment of unbounded variables.

Results (Serial Mode)

	solved	timeout	fail	mean time*
BARON	790	183	27	75s
Lindo API	538	323	139	489s
Octeract	671	279	50	184s
SCIP	776	183	41	85s
virt. worst	368	405	227	1505s
virt. best	967	33	0	20s



*fails are accounted with timelimit

Feedback

(Some of) the competition agrees with this benchmark... :-)

The Optimization Firm @TheOptFirm · 13h

BARON was shown to be the best MINLP solver in a recent study conducted by the SCIP group. Read more: bit.ly/3Hes9PE

BARON

BARON Still Leads Among MINLP Solvers, New Study Shows

The Optimization Firm is excited to share a recent study in which BARON was shown to be the best MINLP solver. The study, which was conducted by one of our top competitors (the SCIP development team), found that BARON still leads in performance compared to the other competing solvers.

1 98

`https://twitter.com/TheOptFirm/status/1616127754033938463`

Parallel Mode

Limiting solvers to one CPU core is so 20th century....

Can these solvers **utilize several cores?** (assuming shared memory)

Parallel Mode

Limiting solvers to one CPU core is so 20th century....

Can these solvers **utilize several cores?** (assuming shared memory)

- BARON can utilize parallelization in the MIP solver, when solving MIP relaxations

Considering only the selected 200 instances without permutations:

	1 thread		4 threads		8 threads		16 threads	
	solved	time	solved	time	solved	time	solved	time
BARON	161	64s	160	58s	160	57s	158	59s

Parallel Mode

Limiting solvers to one CPU core is so 20th century....

Can these solvers **utilize several cores?** (assuming shared memory)

- BARON can utilize parallelization in the MIP solver, when solving MIP relaxations
- Lindo API log says it uses parallellization (parallel tree search?)

Considering only the selected 200 instances without permutations:

	1 thread		4 threads		8 threads		16 threads	
	solved	time	solved	time	solved	time	solved	time
BARON	161	64s	160	58s	160	57s	158	59s
Lindo API	114	424s	114	379s	106	460s	107	456s

Parallel Mode

Limiting solvers to one CPU core is so 20th century....

Can these solvers **utilize several cores?** (assuming shared memory)

- BARON can utilize parallelization in the MIP solver, when solving MIP relaxations
- Lindo API log says it uses parallellization (parallel tree search?)
- Octeract has been designed for parallel tree search from the beginning

Considering only the selected 200 instances without permutations:

	1 thread		4 threads		8 threads		16 threads	
	solved	time	solved	time	solved	time	solved	time
BARON	161	64s	160	58s	160	57s	158	59s
Lindo API	114	424s	114	379s	106	460s	107	456s
Octeract	134	179s	133	147s	138	118s	135	123s

Parallel Mode

Limiting solvers to one CPU core is so 20th century....

Can these solvers **utilize several cores?** (assuming shared memory)

- BARON can utilize parallelization in the MIP solver, when solving MIP relaxations
- Lindo API log says it uses parallellization (parallel tree search?)
- Octeract has been designed for parallel tree search from the beginning
- SCIP itself uses almost no parallelization when solving MINLPs, but FiberSCIP is available (racing ramp-up, parallel tree search)

Considering only the selected 200 instances without permutations:

	1 thread		4 threads		8 threads		16 threads	
	solved	time	solved	time	solved	time	solved	time
BARON	161	64s	160	58s	160	57s	158	59s
Lindo API	114	424s	114	379s	106	460s	107	456s
Octeract	134	179s	133	147s	138	118s	135	123s
(Fiber)SCIP	161	77s	145	94s	147	78s	152	75s

Parallel Mode

Limiting solvers to one CPU core is so 20th century....

Can these solvers **utilize several cores?** (assuming shared memory)

- BARON can utilize parallelization in the MIP solver, when solving MIP relaxations
- Lindo API log says it uses parallellization (parallel tree search?)
- Octeract has been designed for parallel tree search from the beginning
- SCIP itself uses almost no parallelization when solving MINLPs, but FiberSCIP is available (racing ramp-up, parallel tree search)

Considering only the selected 200 instances without permutations:

	1 thread		4 threads		8 threads		16 threads	
	solved	time	solved	time	solved	time	solved	time
BARON	161	64s	160	58s	160	57s	158	59s
Lindo API	114	424s	114	379s	106	460s	107	456s
Octeract	134	179s	133	147s	138	118s	135	123s
(Fiber)SCIP	161	77s	145	94s	147	78s	152	75s

The test set may include too many easy instances where parallelization cannot shine.

End!

For more details, see also

- the report Global Optimization of Mixed-Integer Nonlinear Programs with SCIP 8 (K. Bestuzheva, A. Chmiela, B. Müller, F. Serrano, S. Vigerske, and F. Wegscheider), `https://arxiv.org/abs/2301.00587`, and
- the release report of SCIP 8, `https://arxiv.org/abs/2112.08872`