Solving MINLPs with SCIP

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

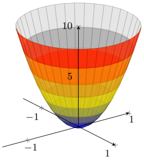
The 22nd Conference of the International Federation of Operational Research Societies
August 27, 2021

ZIB ZUSE INSTITUTE BERLIN
MODAL FORSCHUNGSCAMPUS
SPONSORED BY THE Federal Ministry of Education and Research

Mixed-Integer Nonlinear Programming

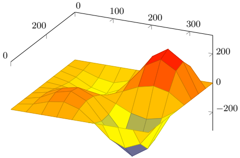
$$\begin{aligned} &\min c^{\mathsf{T}} x \\ &\text{s.t. } g_k(x) \le 0 && \forall k \in [m] \\ &\quad x_i \in \mathbb{Z} && \forall i \in \mathcal{I} \subseteq [n] \\ &\quad x_i \in [\ell_i, u_i] && \forall i \in [n] \end{aligned}$$

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



convex

or



nonconvex

and are given in **algebraic form**.

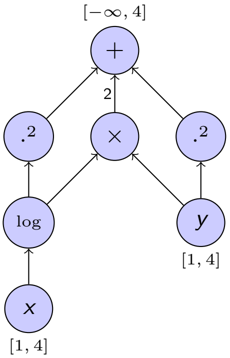
- SCIP solves MINLPs by **spatial Branch & Bound**.

Expression Trees

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

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

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



Expression Trees

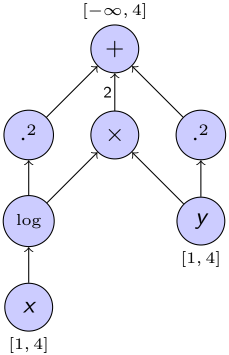
The **algebraic structure** of nonlinear constraints is stored in a directed acyclic graph:

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

Operators:

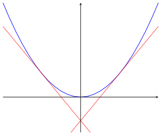
- variable, constant
- $+$, $-$, $*$, $\div$
- $\cdot^2$, $\sqrt{\cdot}$, $\cdot^p$ ($p \in \mathbb{R}$), $x \mapsto x|x|^p$ ($p > 0$)
- $\exp$, $\log$
- abs
- $\sum$, $\prod$
- (user)

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

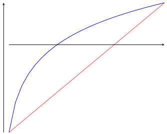


LP relaxation via convexification and linearization:

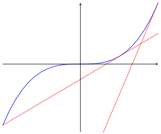
convex functions



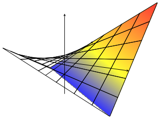
concave functions



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



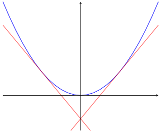
$x \cdot y$



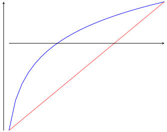
Branch and Bound

LP relaxation via convexification and linearization:

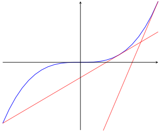
convex functions



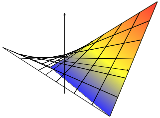
concave functions



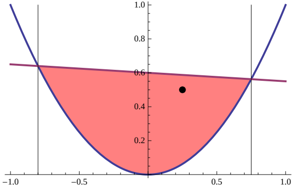
$x^k \quad (k \in 2\mathbb{Z} + 1)$

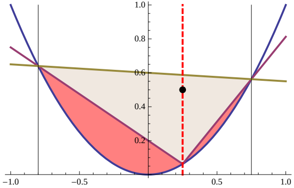


$x \cdot y$



Branching on variables in violated nonconvex constraints:

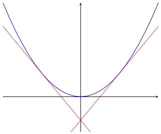




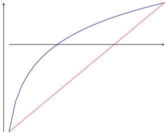
Branch and Bound

LP relaxation via convexification and linearization:

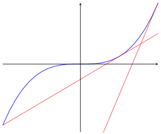
convex functions



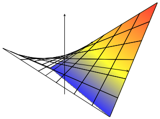
concave functions



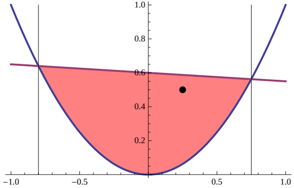
$x^k \quad (k \in 2\mathbb{Z}+1)$

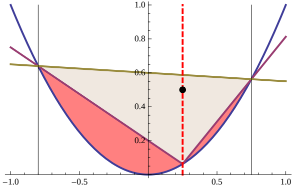


$x \cdot y$



Branching on variables in violated nonconvex constraints:





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

New Framework

Main Ideas

Everything is an expression.

- **ONE** constraint handler: `cons_nonlinear`
- represent all nonlinear constraints in **one expression graph** (DAG)

$$\text{lhs} \leq \text{expression-node} \leq \text{rhs}$$

- all algorithms (check, separation, propagation, etc.) work on the expression graph (no upgrades to specialized nonlinear constraints)

Main Ideas

Everything is an expression.

- **ONE** constraint handler: `cons_nonlinear`
- represent all nonlinear constraints in **one expression graph** (DAG)

$$\text{lhs} \leq \text{expression-node} \leq \text{rhs}$$

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

$\Rightarrow$ **avoid redundancy / ambiguity** of expression types ($\sum$, linear, quadratic, ...)

- stronger identification of **common subexpressions**

Main Ideas

Everything is an expression.

- **ONE** constraint handler: `cons_nonlinear`
- represent all nonlinear constraints in **one expression graph** (DAG)

$$\text{lhs} \leq \text{expression-node} \leq \text{rhs}$$

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

$\Rightarrow$ **avoid redundancy / ambiguity** of expression types ($\sum$, linear, quadratic, ...)

- stronger identification of **common subexpressions**

Do not reformulate constraints.

- introduce **auxiliary variables for the relaxation only**

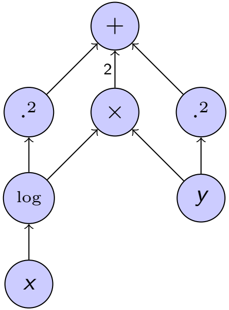
Constraint Enforcement

Constraint:

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

This formulation is used to

- **check feasibility**,
- **presolve**,
- **propagate domains**, ...



Constraint Enforcement

Constraint:

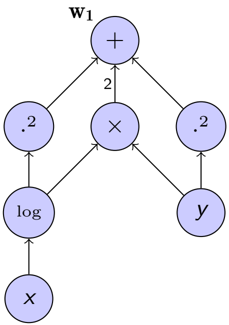
$$\log(x)^2 + 2\log(x)y + y^2 \le 4$$

This formulation is used to

- **check feasibility**,
- **presolve**,
- **propagate domains**, ...

(Implicit) Reformulation:

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



Constraint Enforcement

Constraint:

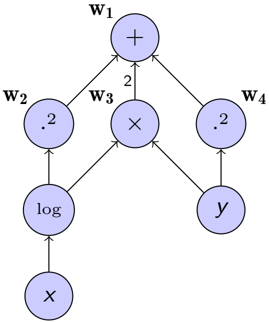
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

- **check feasibility**,
- **presolve**,
- **propagate domains**, ...

(Implicit) Reformulation:

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



Constraint Enforcement

Constraint:

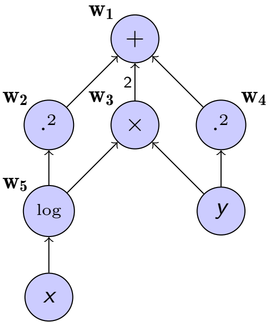
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

- **check feasibility**,
- **presolve**,
- **propagate domains**, ...

(Implicit) Reformulation:

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



Constraint Enforcement

Constraint:

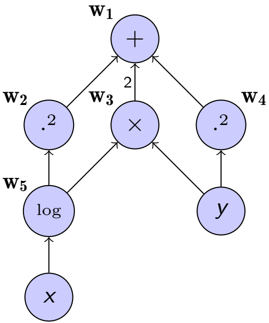
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

- **check feasibility**,
- **presolve**,
- **propagate domains**, ...

(Implicit) Reformulation:

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



Used to **construct LP relaxation**.

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**, thus new ones can be added by users.

Nonlinearity Handlers

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** or constraints.
- Examples: quadratics, convex and concave, second order cone, ...
- At begin of solve or a presolve round, **detection callbacks** are run **only for nodes**
  - that are marked to receive an **auxiliary variable**, or
  - whose bounds will be used, e.g., for **domain propagation**.

Features

- Improved bound propagation for **quadratic** expressions
- **Intersection** cuts
- Separation for 2×2 **principal minors** for constraints $X = xx^T$
- Tight linear relaxations for **second order cones**
- Tight convex relaxations for **bilinear** products
- Reformulation Linearisation Technique cuts for implicit and explicit **bilinear** products
- Tight linear relaxations for **convex** and **concave** expressions
- Generalised **perspective** cuts for functions of semi-continuous variables
- **Symmetry** detection
- Linearization of **products of binary variables**

Performance

MINLPLib, master vs consexpr

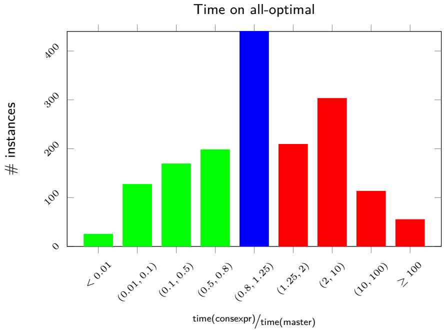
- **master** (classic framework) vs **consexpr** (new framework)
- MINLPLib, all instances that can be **handled by both** master and consexpr
- limits: time = 3600, memory = 50000, gap = 0.0001, absgap = 1e-6
- CPLEX 12.10, Ipopt with MA27, **1 permutation**

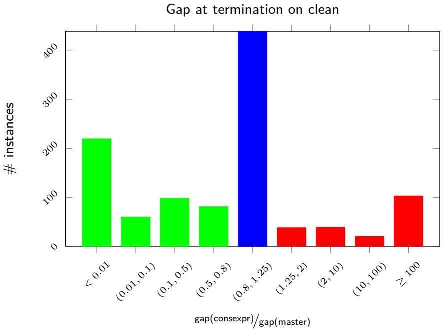
MINLPLib, master vs consexpr

- **master** (classic framework) vs **consexpr** (new framework)
- MINLPLib, all instances that can be **handled by both** master and consexpr
- limits: time = 3600, memory = 50000, gap = 0.0001, absgap = 1e-6
- CPLEX 12.10, Ipopt with MA27, **1 permutation**

	subset	(size)	master	consexpr	master → consexpr	
# fail (checksol=0)	all	(3265)	91	53	79 fixed	41 new fail
# sol. infeas.	all	(3265)	293	31	276 fixed	14 new fail
# solved	clean	(3133)	1933	2044	222 new	111 lost
time (sgm) [s]	all-optimal	(1822)	4.7	5.2	520 faster	680 slower
time (sgm) [s]	all-optimal ∩ [10,tilim)	(640)	61.0	65.9	256 faster	266 slower
time (sgm) [s]	all-optimal ∩ [100,tilim)	(333)	194.2	202.7	142 faster	139 slower
time (sgm) [s]	all-optimal ∩ [1000,tilim)	(119)	837.8	530.0	55 faster	40 slower

MINLPLib, master vs consexpr, time and gap





Coming Soon!

The new framework will be included in the SCIP 8.0 release.