Nonlinear constraints in SCIP

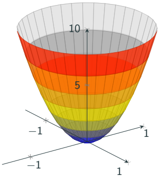
Ksenia Bestuzheva and **Stefan Vigerske**

SCIP Online Workshop 2020 · June 3, 2020

Mixed-Integer Nonlinear Programming

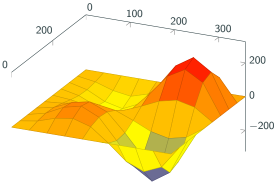
$$
\begin{aligned}
\min\ & c^{\mathsf{T}} x \\
\text{s.t. } & g_k(x) \le 0 && \forall k \in [m] \\
& x_i \in \mathbb{Z} && \forall i \in \mathcal{I} \subseteq [n] \\
& x_i \in [\ell_i, u_i] && \forall i \in [n]
\end{aligned}
$$

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



convex

or



nonconvex

  and are given in algebraic form.

- SCIP solves MINLPs by spatial Branch & Bound.

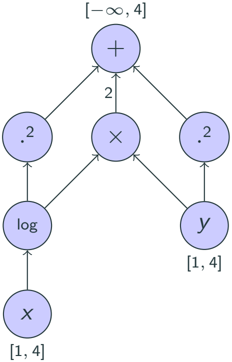
The “classical” framework for (MI)NLP in SCIP

Expression trees and graphs

cons_nonlinear stores algebraic structure of nonlinear constraints in one directed acyclic graph:

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

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



Expression trees and graphs

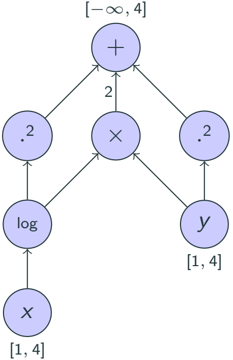
cons_nonlinear stores algebraic structure of nonlinear constraints in one directed acyclic graph:

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

Operators:

- variable index, constant
- $+$, $-$, $*$, $\div$
- $\cdot^2$, $\sqrt{\cdot}$, $\cdot^p$ $(p \in \mathbb{R})$, $\cdot^n$ $(n \in \mathbb{Z})$, $x \mapsto x|x|^{p-1}$ $(p > 1)$
- exp, log
- min, max, abs
- $\sum$, $\prod$, affine-linear, quadratic, signomial
- (user)

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



Expression trees and graphs

cons_nonlinear stores algebraic structure of nonlinear constraints in one directed acyclic graph:

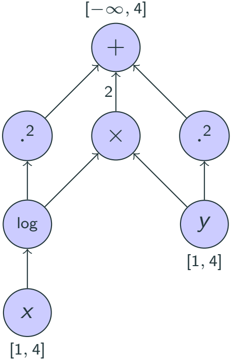
- nodes: variables, operations, constraints
- arcs: flow of computation

Operators:

- variable index, constant
- $+$, $-$, $*$, $\div$
- $\cdot^2$, $\sqrt{\cdot}$, $\cdot^p$ ($p \in \mathbb{R}$), $\cdot^n$ ($n \in \mathbb{Z}$), $x \mapsto x|x|^{p-1}$ ($p > 1$)
- exp, log
- min, max, abs
- $\sum$, $\prod$, affine-linear, quadratic, signomial
- (user)

Additional constraint handler: quadratic, abspower ($x \mapsto x|x|^{p-1}$, $p > 1$), SOC

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

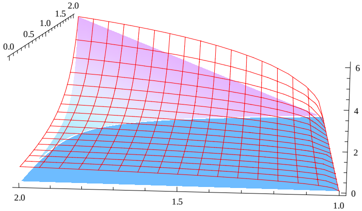


Reformulation in cons_nonlinear (during presolve)

Goal: Reformulate constraints such that only elementary cases (convex, concave, odd power, quadratic) remain.

Example:

$$g(x) = \sqrt{\exp(x_1^2)\ln(x_2)}$$
$$x_1 \in [0,2], \quad x_2 \in [1,2]$$



- reformulates constraints by introducing new variables and new constraints
- other constraint handler can participate

Reformulation in cons_nonlinear (during presolve)

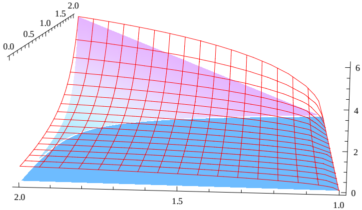
Goal: Reformulate constraints such that only elementary cases (convex, concave, odd power, quadratic) remain.

Example:

$$g(x) = \sqrt{\exp(x_1^2)\ln(x_2)}$$
$$x_1 \in [0,2], \quad x_2 \in [1,2]$$

Reformulation:

$$g = \sqrt{y_1}$$
$$y_1 = y_2 y_3$$
$$y_2 = \exp(y_4)$$
$$y_3 = \ln(x_2)$$
$$y_4 = x_1^2$$



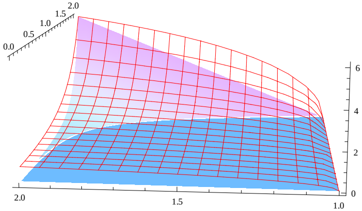
- reformulates constraints by introducing new variables and new constraints
- other constraint handler can participate

Reformulation in cons_nonlinear (during presolve)

Goal: Reformulate constraints such that only elementary cases (convex, concave, odd power, quadratic) remain.

Example:

$$g(x) = \sqrt{\exp(x_1^2)\ln(x_2)}$$
$$x_1 \in [0,2], \quad x_2 \in [1,2]$$



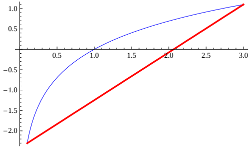
Reformulation:

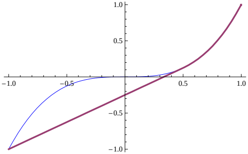
$$\begin{aligned} g &= \sqrt{y_1} & \\ y_1 &= y_2 y_3 & [0, \ln(2)e^4] \\ y_2 &= \exp(y_4) & [1, e^4] \\ y_3 &= \ln(x_2) & [0, \ln(2)] \\ y_4 &= x_1^2 & [0, 4] \end{aligned}$$

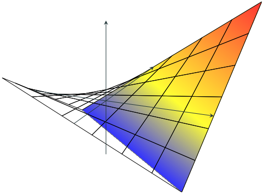
- reformulates constraints by introducing new variables and new constraints
- other constraint handler can participate

Bounding: LP relaxation

- relaxing integrality
- convexifying non-convexities
- linearizing nonlinear convexities



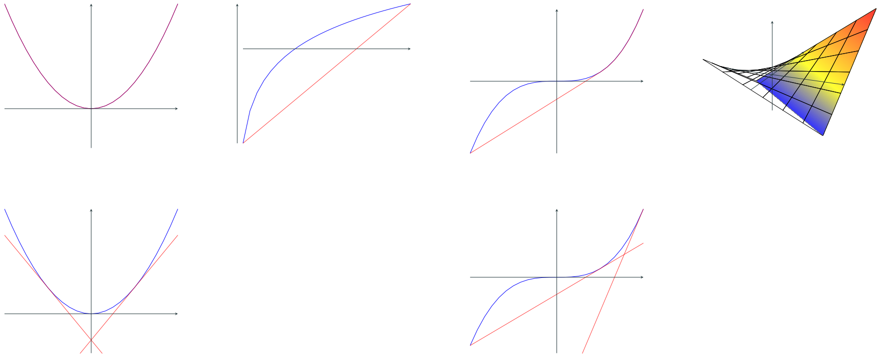




Bounding: LP relaxation

- relaxing integrality
- convexifying non-convexities
- linearizing nonlinear convexities

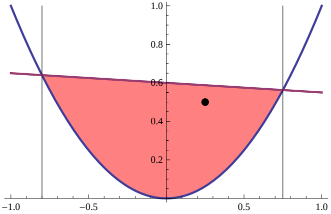
convex functions | concave functions | $x^k \quad (k \in 2\mathbb{Z}+1)$ | $x \cdot y$

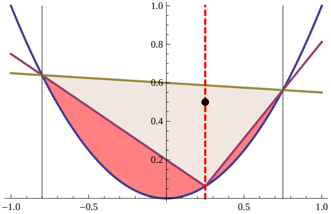


Branching

1. fractional integer variables
2. variables in violated nonconvex constraints, because variable bounds determine the convex relaxation, e.g.,

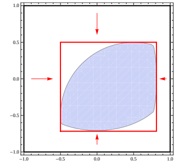
$$x^2 \leq \ell^2 + \frac{u^2 - \ell^2}{u - \ell}(x - \ell) \quad \forall x \in [\ell, u].$$





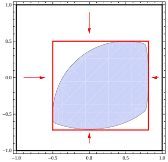
Bound Tightening

- Many constraint handler in SCIP implement a fairly cheap bound tightening (aka. domain propagation) method to infer tighter variable bounds from constraints and current variable bounds.

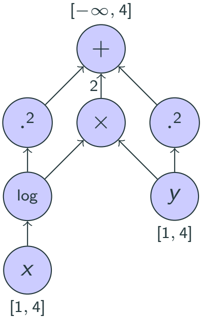


Bound Tightening

- Many constraint handler in SCIP implement a fairly cheap bound tightening (aka. domain propagation) method to infer tighter variable bounds from constraints and current variable bounds.
- cons_nonlinear utilizes the expression graph and interval arithmetic

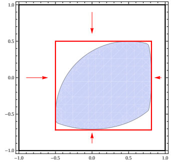


$$\log(x)^2 + 2\log(x)y + y^2 \in [-\infty, 4]$$
$$x, y \in [1, 4]$$

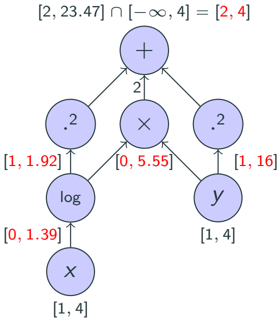


Bound Tightening

- Many constraint handler in SCIP implement a fairly cheap bound tightening (aka. domain propagation) method to infer tighter variable bounds from constraints and current variable bounds.
- cons_nonlinear utilizes the expression graph and interval arithmetic
- Forward propagation:
    - compute bounds on intermediate nodes (top-down)

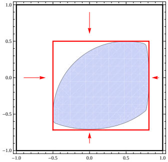


$$\log(x)^2 + 2\log(x)y + y^2 \in [-\infty, 4]$$
$$x, y \in [1, 4]$$

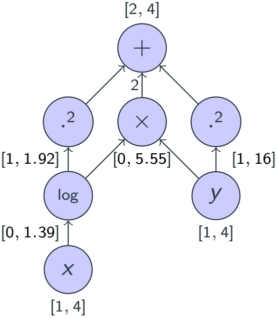


Bound Tightening

- Many constraint handler in SCIP implement a fairly cheap bound tightening (aka. domain propagation) method to infer tighter variable bounds from constraints and current variable bounds.
- cons_nonlinear utilizes the expression graph and interval arithmetic
- Forward propagation:
  - compute bounds on intermediate nodes (top-down)
- Backward propagation:
  - reduce bounds using reverse operations (bottom-up)

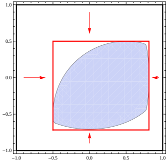


$$\log(x)^2 + 2\log(x)y + y^2 \in [-\infty, 4]$$
$$x, y \in [1, 4]$$

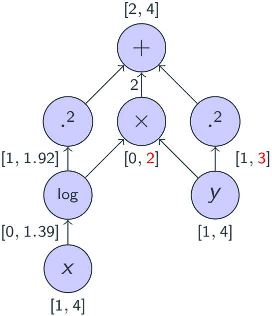


Bound Tightening

- Many constraint handler in SCIP implement a fairly cheap bound tightening (aka. domain propagation) method to infer tighter variable bounds from constraints and current variable bounds.
- cons_nonlinear utilizes the expression graph and interval arithmetic
- Forward propagation:
  - compute bounds on intermediate nodes (top-down)
- Backward propagation:
  - reduce bounds using reverse operations (bottom-up)

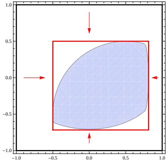


$$\log(x)^2 + 2\log(x)y + y^2 \in [-\infty, 4]$$
$$x, y \in [1, 4]$$

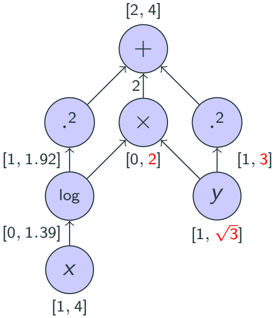


Bound Tightening

- Many constraint handler in SCIP implement a fairly cheap bound tightening (aka. domain propagation) method to infer tighter variable bounds from constraints and current variable bounds.
- cons_nonlinear utilizes the expression graph and interval arithmetic
- Forward propagation:
  - compute bounds on intermediate nodes (top-down)
- Backward propagation:
  - reduce bounds using reverse operations (bottom-up)



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



Further Techniques

Primal Heuristics:

- NLP solving: subnlp, nlpdiving, multistart, mpec
- MINLP solving: LNS heuristics (RENS, RINS, DINS, etc.)
- MIP solving: undercover

Further Techniques

Primal Heuristics:

- NLP solving: subnlp, nlpdiving, multistart, mpec
- MINLP solving: LNS heuristics (RENS, RINS, DINS, etc.)
- MIP solving: undercover

Tighter Relaxations:

- second-order cone upgrade of quadratic constraints
- adding KKT reformulation (using SOS1) for QPs
- $x \cdot y$ over 2D projections of the LP relaxation
- separation for edge-concave quadratic constraints (off by default)
- projection of LP relaxation onto convex feasible sets (off by default)

Further Techniques

Primal Heuristics:

- NLP solving: subnlp, nlpdiving, multistart, mpec
- MINLP solving: LNS heuristics (RENS, RINS, DINS, etc.)
- MIP solving: undercover

Tighter Relaxations:

- second-order cone upgrade of quadratic constraints
- adding KKT reformulation (using SOS1) for QPs
- $x \cdot y$ over 2D projections of the LP relaxation
- separation for edge-concave quadratic constraints (off by default)
- projection of LP relaxation onto convex feasible sets (off by default)

More Bound Tightening:

- Optimization-Based Bound Tightening: min / max x_i w.r.t. LP relaxation
- nl. Optimization-Based Bound Tightening: min / max x_i w.r.t. convex NLP relaxation (off by default)

Interfaces

A MINLP can be input via

- File readers: FlatZinc*, LP*, MPS*, OSiL, PIP†, ZIMPL
- Interfaces: AMPL, C, GAMS, Java*, Julia/JuMP, Matlab (via OPTI Toolbox), Python

SCIP can utilize this software for MINLP solving:

- NLP Solvers: Ipopt, FilterSQP, WORHP
- Automatic Differentiation: CppAD

---

* quadratic only

† polynomial only

Example: Circle Packing

A new framework for NLP in SCIP (work in progress)

by K. Bestuzheva, B. Müller, F. Serrano, S. Vigerske, F. Wegscheider

Problem with current implementation

Consider

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

An optimal solution:

$x = -1$

$y = 1$

$z = 1000$

Problem with current implementation

Consider

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

An optimal solution:

$x = -1$

$y = 1$

$z = 1000$

SCIP reports

```
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 %
  [nonlinear] <e1>: exp((7.90775527898213681 +1 (<x> * <y>)))-1<z>[C]  <= 0;
violation: right hand side is violated by 0.000673453314561812
best solution is not feasible in original problem

x                                    -1.00057454873626  (obj:0)
y                                    0.999425451364613  (obj:0)
z                                     999.999656552061  (obj:1)
```

Reformulated problem

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

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

Reformulated problem

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

$$\begin{aligned} &\min z & &\text{Violation} \\ &\text{s.t. } \exp(w) \le z & &0.4659 \cdot 10^{-6} \le \texttt{numerics/feastol} \checkmark \\ &\quad \ln(1000) + 1 + x\,y = w & &0.6731 \cdot 10^{-6} \le \texttt{numerics/feastol} \checkmark \\ &\quad x^2 + y^2 \le 2 & &0.6602 \cdot 10^{-6} \le \texttt{numerics/feastol} \checkmark \end{aligned}$$

Solution (`found by <relaxation>`):

x = -1.000574549
y = 0.999425451
z = 999.999656552
w = 6.907754936

Reformulated problem

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

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

Solution (`found by <relaxation>`):

x = -1.000574549
y = 0.999425451
z = 999.999656552
w = 6.907754936

$\Rightarrow$ Explicit reformulation of constraints …

- … loses the connection to the original problem.

Reformulated problem

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

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

Solution (`found by <relaxation>`):

x = -1.000574549
y = 0.999425451
z = 999.999656552
w = 6.907754936

$\Rightarrow$ 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.

Main Ideas

Everything is an expression.

- **ONE** constraint handler: `cons_expr`
- 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_expr`
- 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, etc.)

$\Rightarrow$ avoid redundancy / ambiguity of expression types (classic: $+$, $\sum$, linear, quad., ...)

- stronger identification of common subexpressions

Main Ideas

Everything is an expression.

- **ONE** constraint handler: `cons_expr`
- 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, etc.)

$\Rightarrow$ avoid redundancy / ambiguity of expression types (classic: $+$, $\sum$, linear, quad., ...)

- stronger identification of common subexpressions

Do not reformulate constraints.

- introduce auxiliary variables for the relaxation only

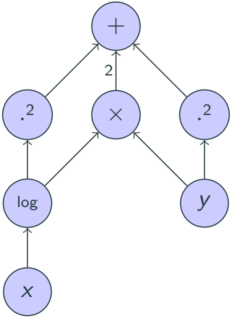
Enforcement

Constraint:

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

This formulation is used to

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



Enforcement

Constraint:

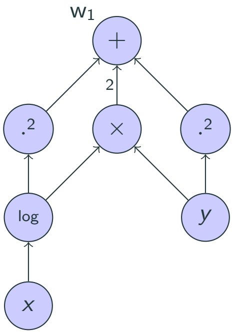
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

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

(Implicit) Reformulation:

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



Enforcement

Constraint:

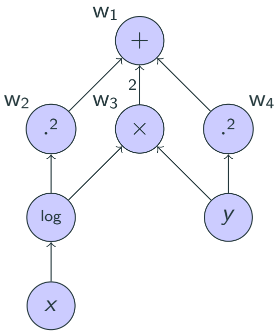
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

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

(Implicit) Reformulation:

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



Enforcement

Constraint:

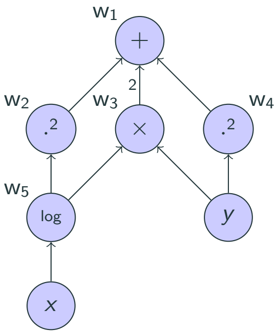
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

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

(Implicit) Reformulation:

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



Enforcement

Constraint:

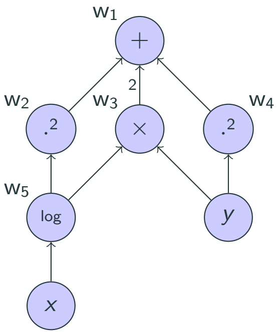
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

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 handler

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 handler are like other SCIP plugins, thus new ones can be added by users.

Motivating example revisited

$$\min z \quad \text{s.t. } \exp(\ln(1000)+1+x\,y) \leq z,\; x^2+y^2 \leq 2$$

Classic:

```
presolving (5 rounds: 5 fast, 1 medium, 1 exhaustive):
 0 deleted vars, 0 deleted constraints, 1 added constraints,...
 0 implications, 0 cliques
presolved problem has 4 variables (0 bin, 0 int, 0 impl, 4 cont)
  and 3 constraints
      2 constraints of type <quadratic>
      1 constraints of type <nonlinear>

[...]

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 %
  [nonlinear] <e1>: exp((7.90776 + (<x> * <y>)))-1<z>[C] <= 0;
violation: right hand side is violated by 0.000673453314561812
best solution is not feasible in original problem

x                                   -1.00057454873626  (obj:0)
y                                   0.999425451364613  (obj:0)
z                                    999.999656552061  (obj:1)
```

Motivating example revisited

$$\min z \quad \text{s.t. } \exp(\ln(1000) + 1 + x\,y) \leq z,\; x^2 + y^2 \leq 2$$

Classic:

```
presolving (5 rounds: 5 fast, 1 medium, 1 exhaustive):
 0 deleted vars, 0 deleted constraints, 1 added constraints,...
 0 implications, 0 cliques
presolved problem has 4 variables (0 bin, 0 int, 0 impl, 4 cont)
  and 3 constraints
      2 constraints of type <quadratic>
      1 constraints of type <nonlinear>

[...]

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 %
  [nonlinear] <e1>: exp((7.90776 + (<x> * <y>)))-1<z>[C] <= 0;
violation: right hand side is violated by 0.000673453314561812
best solution is not feasible in original problem

x                                   -1.00057454873626   (obj:0)
y                                   0.999425451364613   (obj:0)
z                                    999.999656552061   (obj:1)
```

New:

```
presolving (3 rounds: 3 fast, 1 medium, 1 exhaustive):
 0 deleted vars, 0 deleted constraints, 0 added constraints,...
 0 implications, 0 cliques
presolved problem has 3 variables (0 bin, 0 int, 0 impl, 3 cont)
  and 2 constraints
      2 constraints of type <expr>

[...]

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.47
Solving Nodes      : 15
Primal Bound       : +9.99999949950021e+02 (2 solutions)
Dual Bound         : +9.99999949950021e+02
Gap                : 0.00 %

x                                   -1.00000002499999   (obj:0)
y                                    1.00000002499999   (obj:0)
z                                    999.999949950021   (obj:1)
```

Exploiting structure

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

Smarter reformulation:

- Recognize that $\log(x)^2 + 2\log(x)y + y^2$ is convex in $(\log(x), y)$.

Exploiting structure

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

Smarter reformulation:

- Recognize that $\log(x)^2 + 2\log(x)y + y^2$ is convex in $(\log(x), y)$.

$\Rightarrow$ Introduce auxiliary variable for $\log(x)$ only.

$$\begin{aligned} w^2 + 2wy + y^2 &\leq 4 \\ \log(x) &= w \end{aligned}$$

Handle $w^2 + 2wy + y^2 \leq 4$ as convex constraint ("gradient-cuts").

Exploiting structure

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

Smarter reformulation:

- Recognize that $\log(x)^2 + 2\log(x)y + y^2$ is convex in $(\log(x), y)$.

$\Rightarrow$ Introduce auxiliary variable for $\log(x)$ only.

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

Handle $w^2 + 2wy + y^2 \leq 4$ as convex constraint ("gradient-cuts").

Nonlinearity Handler (nlhdlrs):

- Adds additional separation and 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 subexpressions, vertex-polyhedral

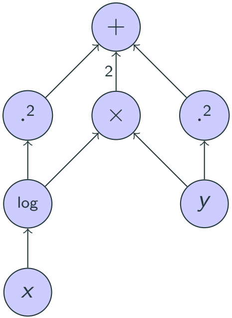
Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

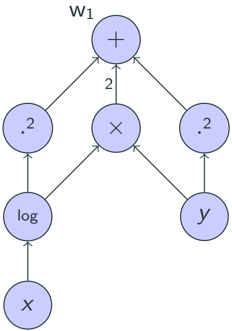


Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

$$w_1 \leq 4$$



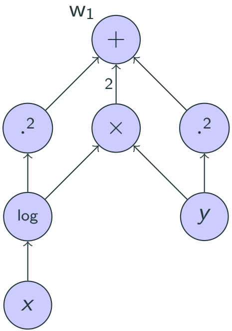
1. Add auxiliary variable w_1 for root.

Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

$$w_1 \leq 4$$



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

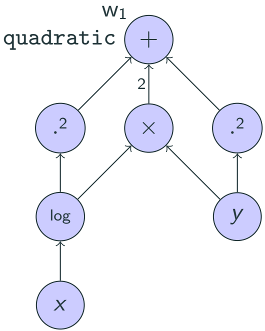
Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

$$w_1 \leq 4$$

1. Add auxiliary variable w_1 for root.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_quadratic` detects a convex quadratic structure and signals success.

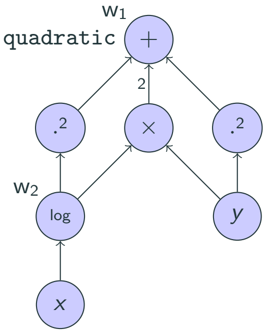


Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

$$w_1 \leq 4$$
$$w_2^2 + 2w_2 y + y^2 \leq w_1 \quad [\texttt{nlhdlr\_quadratic}]$$



1. Add auxiliary variable w_1 for root.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_quadratic` detects a convex quadratic structure and signals success.
   - `nlhdlr_quadratic` adds an auxiliary variable w_2 for log node.

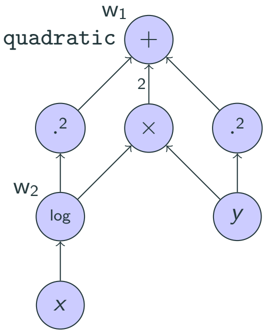
Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

$$w_1 \leq 4$$

$$w_2^2 + 2w_2 y + y^2 \leq w_1 \quad [\texttt{nlhdlr\_quadratic}]$$



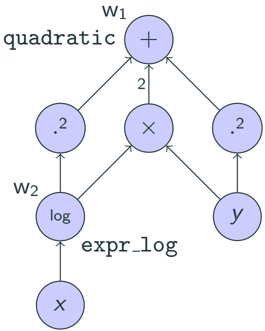
1. Add auxiliary variable w_1 for root.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_quadratic` detects a convex quadratic structure and signals success.
   - `nlhdlr_quadratic` adds an auxiliary variable w_2 for log node.
3. Run `detect` of all nlhdlrs on log node.

Nonlinearity Handler in Expression Graph

- Nodes in the expression graph can have one or several nlhdlrs attached.
- At beginning of solve, detection callbacks are run **only** for nodes that have auxiliary variable. Detection callback may add auxiliary variables.

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

$$
\begin{aligned}
w_1 &\le 4 \\
w_2^2 + 2w_2 y + y^2 &\le w_1 \quad [\texttt{nlhdlr\_quadratic}] \\
\log(x) &= w_2 \quad [\texttt{expr\_log}]
\end{aligned}
$$



1. Add auxiliary variable w_1 for root.
2. Run `detect` of all nlhdlrs on $+$ node.
   - `nlhdlr_quadratic` detects a convex quadratic structure and signals success.
   - `nlhdlr_quadratic` adds an auxiliary variable w_2 for log node.
3. Run `detect` of all nlhdlrs on log node.
   - No specialized nlhdlr signals success. The expression handler will be used.

Current status

Available features:

- Handler for quadratic subexpressions
- Handler for second-order cone structures
- Handler for convex and concave subexpressions
- Handler for functions on semi-continuous variables (perspective formulations)
- Handler for bilinear terms ($x \cdot y$ over 2D projection of LP relaxation)
- RLT (Reformulation-Linearization Technique) separator for bilinear terms
- Separator for SDP cuts on 2×2 principal minors of $X - xx^\mathsf{T} \succeq 0$
- Linearization of products of binary variables
- Symmetry detection
- Support for operators cos, sin, entropy
- ...

Current status

Available features:

- Handler for quadratic subexpressions
- Handler for second-order cone structures
- Handler for convex and concave subexpressions
- Handler for functions on semi-continuous variables (perspective formulations)
- Handler for bilinear terms ($x \cdot y$ over 2D projection of LP relaxation)
- RLT (Reformulation-Linearization Technique) separator for bilinear terms
- Separator for SDP cuts on 2×2 principal minors of $X - xx^\top \succeq 0$
- Linearization of products of binary variables
- Symmetry detection
- Support for operators cos, sin, entropy
- ...

Main open tasks before release:

- check performance "outliers"
- replacing remaining classic code by new one (in particular NLP relaxation and NLP solver interfaces)
- documentation "?"

Nonlinear constraints in SCIP

Ksenia Bestuzheva and **Stefan Vigerske**

SCIP Online Workshop 2020 · June 3, 2020