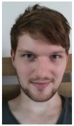
Revising the handling of nonlinear constraints in SCIP

Work in Progress Report

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





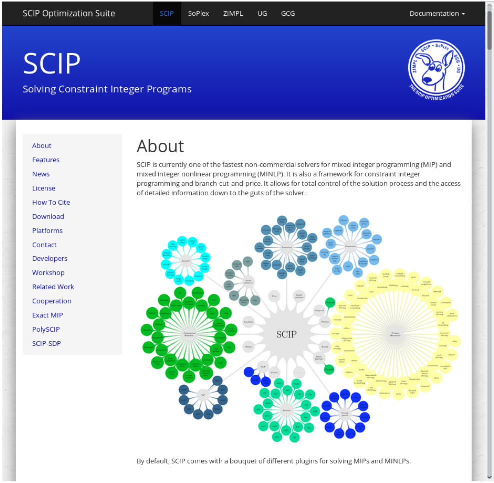


WCGO · Metz, France · July 9, 2019

SCIP: Solving Constraint Integer Programs

- modular branch-cut-and-price framework for constraint integer programming
- includes full-fledged MIP/MINLP solver
- part of SCIP Optimization Suite (GCG, SCIP, SoPlex, UG, ZIMPL)
- Latest Release Report: The SCIP Optimization Suite 6.0 by Gleixner, Bastubbe, Eifler, Gally, Gamrath, Gottwald, Hendel, Hojny, Koch, Lübbecke, Maher, Miltenberger, Müller, Pfetsch, Puchert, Rehfeldt, Schlösser, Schubert, Serrano, Shinano, Viernickel, Wegscheider, Witt, Witzig
- free for academic use

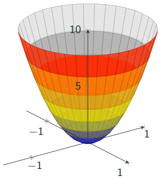
Download at `scip.zib.de`:



Mixed-Integer Nonlinear Programming

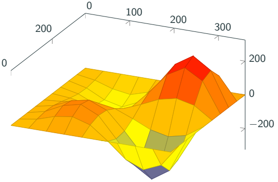
$$
\begin{aligned}
\min\ & c^{\mathsf{T}} x \\
\text{s.t. } & g_k(x) \leq 0 && \forall k \in [m] \\
& x_i \in \mathbb{Z} && \forall i \in \mathcal{I} \subseteq [n] \\
& x_i \in [\ell_i, u_i] && \forall i \in [n]
\end{aligned}
$$

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



convex

or



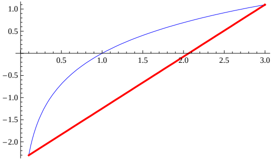
nonconvex

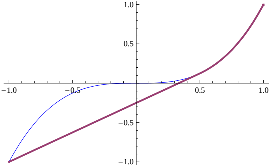
and are given in algebraic form.

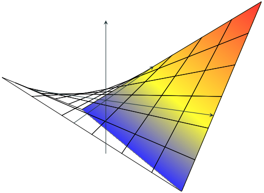
SCIP solves MINLPs by spatial Branch & Bound

Ingredients:

- constructing an LP relaxation by
  - relaxing integrality
  - convexifying non-convexities



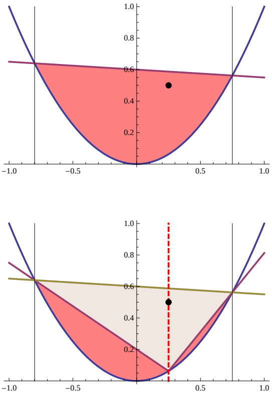




SCIP solves MINLPs by spatial Branch & Bound

Ingredients:

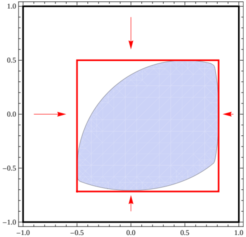
- constructing an LP relaxation by
  - relaxing integrality
  - convexifying non-convexities
- branching on
  - fractional integer variables
  - variables in violated nonconvex constraints

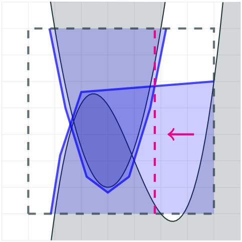


SCIP solves MINLPs by spatial Branch & Bound

Ingredients:

- constructing an LP relaxation by
  - relaxing integrality
  - convexifying non-convexities
- branching on
  - fractional integer variables
  - variables in violated nonconvex constraints
- tightening of variable bounds (domain propagation)
- primal heuristics
- presolving / reformulation





Current Implementation (SCIP 6.0)

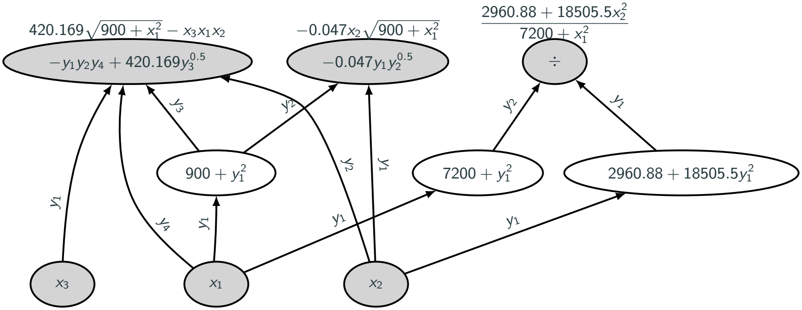
Expression trees and graphs

cons_nonlinear ($\text{lhs} \leq \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} c_j f_j(x) \leq \text{rhs}$) stores the nonlinear functions f_j of *all* constraints in one expression graph (DAG).

For example (MINLPLib instance `nvs01`):

$$420.169\sqrt{900 + x_1^2} - x_3 x_1 x_2 = 0 \qquad \frac{2960.88 + 296088 \cdot 0.0625 x_2^2}{7200 + x_1^2} - x_3 \geq 0$$

$$x_{\text{obj}} - 0.047 x_2 \sqrt{900 + x_1^2} \geq 0$$



- some use of common subexpression

Expression operators and constraint handler

Operators (handled by cons_nonlinear):

- 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)

Expression operators and constraint handler

Operators (handled by cons_nonlinear):

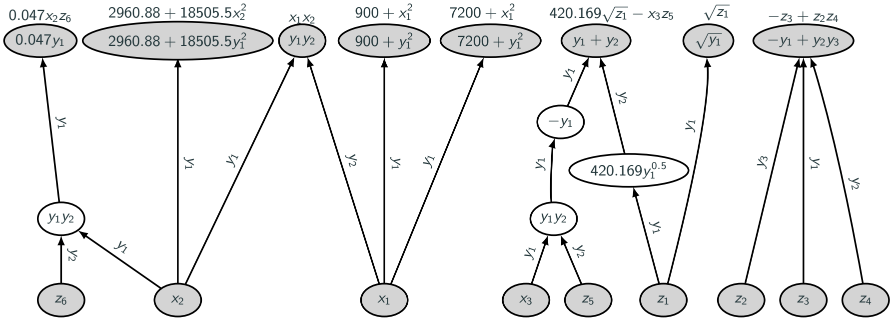
- variable index, constant
- $+$, $-$, $*$, $\div$
- $\cdot^2$, $\sqrt{\cdot}$, $\cdot^p$ ($p \in \mathbb{R}$), $\cdot^n$ ($n \in \mathbb{Z}$), $x \mapsto x|x|^{p-1}$ ($p > 1$)
- exp, log
- min, max, abs
- $\sum$, $\prod$, affine-linear, quadratic, signomial
- (user)

Additional constraint handler:

- quadratic
- abspower ($x \rightarrow x|x|^{p-1}$, $p > 1$)
- SOC (second-order cones)
- (bivariate)

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

$$900 + x_1^2 = z_1 \qquad 7200 + x_1^2 = z_4 \qquad 420.169\sqrt{z_1} - x_3 z_5 = 0$$
$$-z_3 + z_2 z_4 = 0 \qquad x_1 x_2 = z_5 \qquad z_2 - x_3 \geq 0$$
$$2960.88 + 18505.5 x_2^2 = z_3 \qquad \sqrt{z_1} = z_6 \qquad 0.047 x_2 z_6 \leq x_{obj}$$



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

Problem with this approach

Consider

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

An optimal solution:

$x = -1$

$y = 1$

$z = 1000$

Problem with this approach

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

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

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

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 ...

- ... looses 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. $\exp(w) \leq z$	$0.4659 \cdot 10^{-6} \leq$ `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

⇒ Explicit reformulation of constraints ...

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

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

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

Fundamental structure

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.

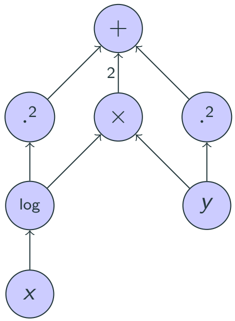
- introduce auxiliary variables for the relaxation only

Constraint:

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

This formulation is used to

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



Enforcement

Constraint:

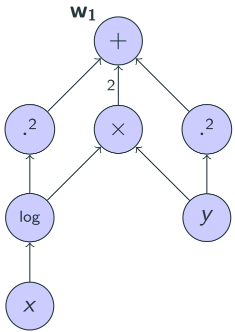
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

This formulation is used to

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

(Implicit) Reformulation:

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



Constraint:

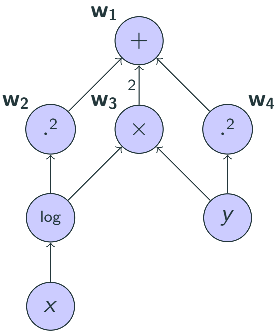
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

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:

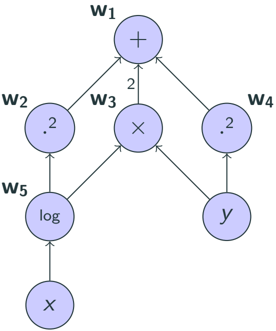
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

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:

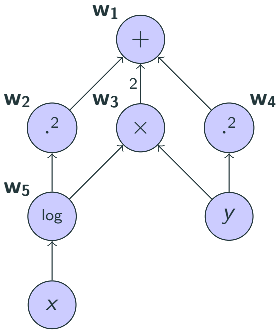
$$\log(x)^2 + 2\log(x)y + y^2 \leq 4$$

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
- distribute branching scores to operands
- 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

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

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)
```

Performance

- Testset: 1618 instances from MINLPLib[1]
- Time limit: 30 minutes, Optimality gap tolerance: 0.01%
- LP solver: CPLEX 12.9.0.0, NLP solver: IPOPT 3.12.11

	classic code	new code
# solution infeasible	90	9

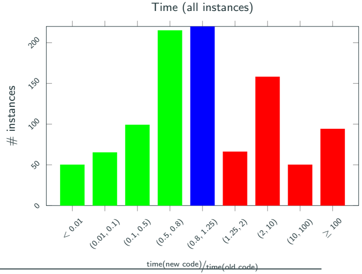
[1] `http://www.minlplib.org`, currently 1626 instances

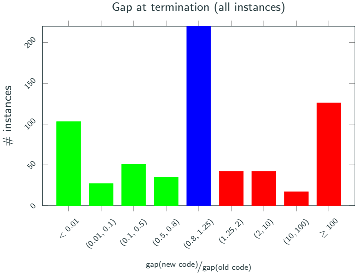
[2] shifted geometric mean with shift = 1s: $\prod_{i=1}^{n}(t_i + 1)^{1/n} - 1$

Performance

- Testset: 1618 instances from MINLPLib[1]
- Time limit: 30 minutes, Optimality gap tolerance: 0.01%
- LP solver: CPLEX 12.9.0.0, NLP solver: IPOPT 3.12.11

	classic code	new code
# solution infeasible	90	9
# solved (out of 1618)	827	807
# solved by both	701	
mean time[2] (on solved by both)	3.73s	4.52s





[1] http://www.minlplib.org, currently 1626 instances

[2] shifted geometric mean with shift = 1s: $\prod_{i=1}^{n}(t_i + 1)^{1/n} - 1$

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

Acceleration

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:

- 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

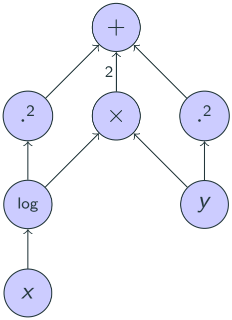
Nonlinearity Handler in Expression Graph

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



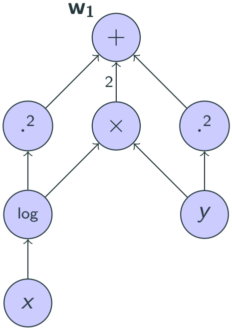
Nonlinearity Handler in Expression Graph

- 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.



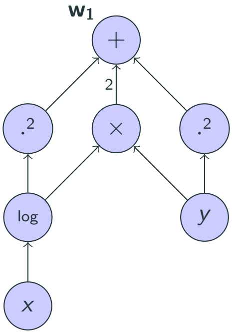
Nonlinearity Handler in Expression Graph

- 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.



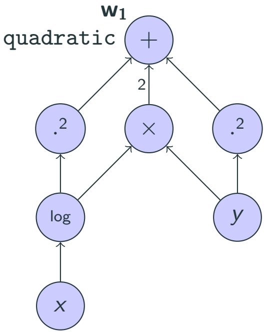
Nonlinearity Handler in Expression Graph

- 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.



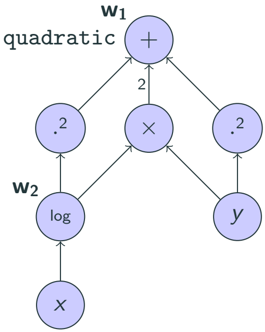
Nonlinearity Handler in Expression Graph

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

$$w_1 \leq 4$$

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



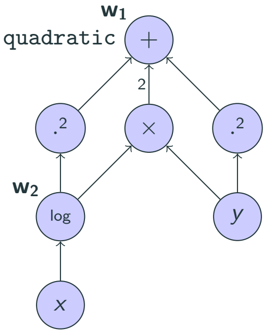
1. Add auxiliary variable w_1 for root.
2. Run `detect` of all nlhdlrs on + node.
   - `nlhdlr_quadratic` detects a convex quadratic structure and signals success.
   - `nlhdlr_quadratic` adds an auxiliary variable w_2 for log node.

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

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



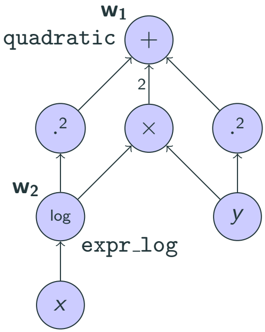
1. Add auxiliary variable w_1 for root.
2. Run `detect` of all nlhdlrs on + node.
   - `nlhdlr_quadratic` detects a convex quadratic structure and signals success.
   - `nlhdlr_quadratic` adds an auxiliary variable w_2 for log node.
3. Run `detect` of all nlhdlrs on log node.

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

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



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.

Handler for quadratic subexpressions

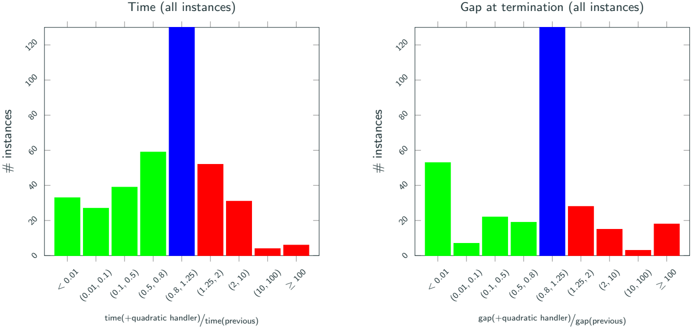
- Recognize quadratic forms (sums of squares and products in two terms).
- Recognize convexity by checking coefficient matrix for positive semidefiniteness. Use this to provide tight linear underestimators by linearization.
- Provide better bound tightening, in particular for univariate quadratics:

$$\{ax^2 + bx : x \in [\ell, u]\} = \begin{cases} \operatorname{conv}\{a\ell^2 + b\ell, au^2 + bu, -\frac{b^2}{4a}\}, & \text{if } -\frac{b}{2a} \in [\ell, u], \\ \operatorname{conv}\{a\ell^2 + b\ell, au^2 + bu\}, & \text{otherwise} \end{cases}$$

$$\{x : ax^2 + bx \geq c\} = \begin{cases} \left[-\infty, -\sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}\right] \cup \left[\sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}, \infty\right], & \text{if } a > 0, \\ \left[-\sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}, \sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}\right], & \text{if } a < 0. \end{cases}$$

Impact of handler for quadratics

	previous (base case)	+ quadratic handler
# solved (out of 1618)	807	843
# solved by both	790	
# solved by both and affected[3]	312	
mean time[4] (on solved&affected)	12.7s	9.7s

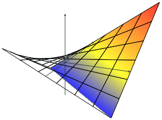


[3] affected = different search path, indicated by different number of B&B nodes or LP iterations
[4] shifted geometric mean with shift = 1s: $\prod_{i=1}^{n}(t_i + 1)^{1/n} - 1$

Separator for RLT

- for bilinear products $x_i x_j$, we may have introduced auxiliary variables $w_{i,j}$
- the expression handler for products generates McCormick inequalities:

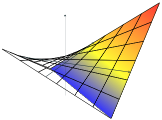
$$\begin{aligned}
(x_i - \ell_i)(x_j - \ell_j) \geq 0 \quad &\Rightarrow \quad w_{i,j} \geq \ell_i x_j + \ell_j x_i - \ell_i \ell_j \\
(x_i - u_i)(x_j - u_j) \geq 0 \quad &\Rightarrow \quad w_{i,j} \geq u_i x_j + u_j x_i - u_i u_j \\
(x_i - \ell_i)(x_j - u_j) \leq 0 \quad &\Rightarrow \quad w_{i,j} \leq \ell_i x_j + u_j x_i - \ell_i u_j \\
(x_i - u_i)(x_j - \ell_j) \leq 0 \quad &\Rightarrow \quad w_{i,j} \leq u_i x_j + \ell_j x_i - u_i \ell_j
\end{aligned}$$



Separator for RLT

- for bilinear products $x_i x_j$, we may have introduced auxiliary variables $w_{i,j}$
- the expression handler for products generates McCormick inequalities:

$$
\begin{aligned}
(x_i - \ell_i)(x_j - \ell_j) \geq 0 \quad &\Rightarrow \quad w_{i,j} \geq \ell_i x_j + \ell_j x_i - \ell_i \ell_j \\
(x_i - u_i)(x_j - u_j) \geq 0 \quad &\Rightarrow \quad w_{i,j} \geq u_i x_j + u_j x_i - u_i u_j \\
(x_i - \ell_i)(x_j - u_j) \leq 0 \quad &\Rightarrow \quad w_{i,j} \leq \ell_i x_j + u_j x_i - \ell_i u_j \\
(x_i - u_i)(x_j - \ell_j) \leq 0 \quad &\Rightarrow \quad w_{i,j} \leq u_i x_j + \ell_j x_i - u_i \ell_j
\end{aligned}
$$



Reformulation-Linearization Technique [Adams and Sherali, 1986]:

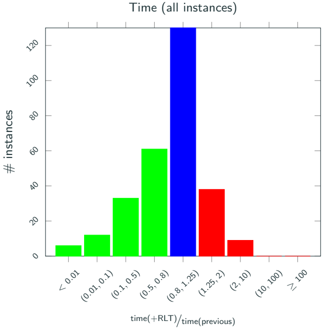
- additional valid cuts can be obtained by multiplication with linear constraints:

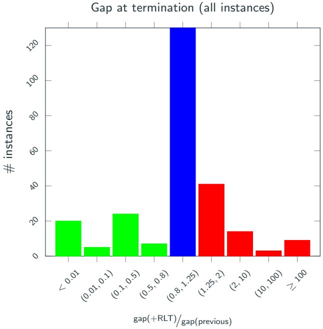
$$
\begin{aligned}
a^{\mathsf{T}} x \geq b \quad \times \quad x_j - \ell_j \quad &\Rightarrow \quad a^{\mathsf{T}} w_{\cdot,j} - a^{\mathsf{T}} x\, \ell_j \geq b x_j - b\, \ell_j \\
a^{\mathsf{T}} x = b \quad \times \quad x_j \quad &\Rightarrow \quad a^{\mathsf{T}} w_{\cdot,j} = b x_j
\end{aligned}
$$

- in our implementation, we only look for RLT cuts that do not introduce new auxiliary variables $w_{i,j}$
- very effective for pooling problems

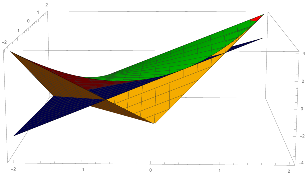
Impact of RLT separator

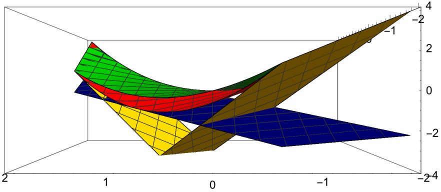
	previous	+ RLT
# solved (out of 1618)	843	857
# solved by both	834	
# solved by both and affected	125	
mean time (on solved&affected)	7.5s	5.0s





- McCormick inequalities give convex hull for $x_i x_j$ on box $[\ell_i, \ell_j] \times [u_i, u_j]$
- they do not if additional inequalities are present, e.g., $x_i \leq x_j$:





green — graph of $w_{ij} = x_i x_j$

yellow — McCormick relaxation of $x_i x_j$ over $[-2, 2]^2$

red — convex envelope of $x_i x_j$ over $\{(x_i, x_j) \in [-2, 2]^2 : x_i \leq x_j\}$

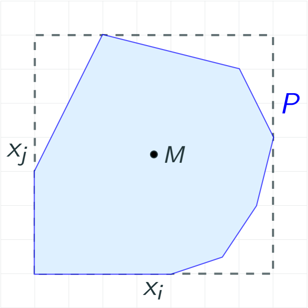
- closed formulas and algorithms are known [Linderoth 2004, Hijazi 2015, Locatelli 2016]

2D projections for $x_i x_j$

Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i, x_j}(\text{LP})$

- assume variable bounds are tight
- $M := (\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}) \in P$
- every facet of P separates at most one of the 4 corners



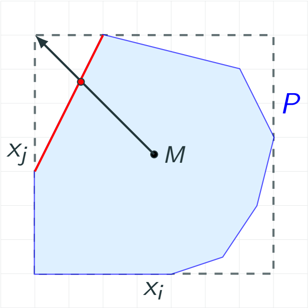
[5]Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

2D projections for $x_i x_j$

Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i,x_j}(\text{LP})$

- assume variable bounds are tight
- $M := (\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}) \in P$
- every facet of P separates at most one of the 4 corners
- optimize along directions from M to each corner



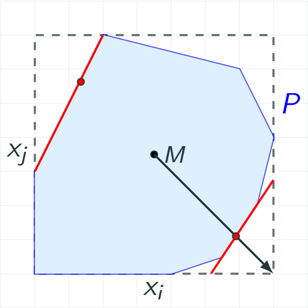
[5]Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

2D projections for $x_i x_j$

Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i,x_j}(\text{LP})$

- assume variable bounds are tight
- $M := (\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}) \in P$
- every facet of P separates at most one of the 4 corners
- optimize along directions from M to each corner



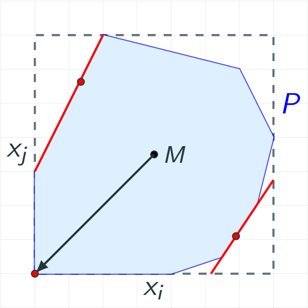
[5] Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

2D projections for $x_i x_j$

Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i,x_j}(\text{LP})$

- assume variable bounds are tight
- $M := (\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}) \in P$
- every facet of P separates at most one of the 4 corners
- optimize along directions from M to each corner



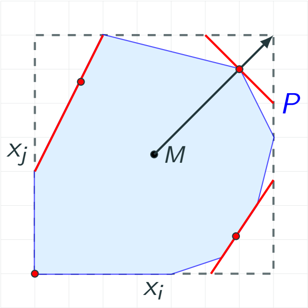
[5] Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

2D projections for $x_i x_j$

Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i,x_j}(\text{LP})$

- assume variable bounds are tight
- $M := (\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}) \in P$
- every facet of P separates at most one of the 4 corners
- optimize along directions from M to each corner



[5]Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

2D projections for $x_i x_j$

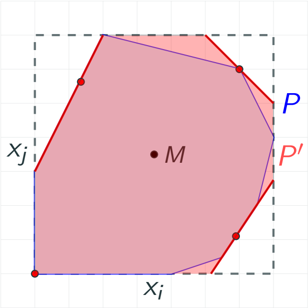
Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i,x_j}(\text{LP})$

- assume variable bounds are tight
- $M := (\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}) \in P$
- every facet of P separates at most one of the 4 corners
- optimize along directions from M to each corner

$\Rightarrow$ $P' \supseteq P$ described by at most

- 4 nontrivial inequalities
- 4 axis-parallel inequalities



[5]Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

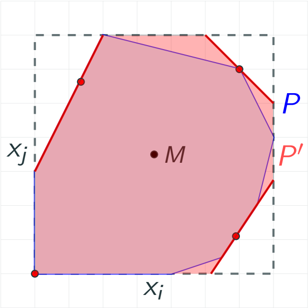
2D projections for $x_i x_j$

Problem: inequalities utilizing only x_i and x_j may not be present in problem

Solution[5]: Project LP relaxation onto (x_i, x_j), $P := \text{proj}_{x_i,x_j}(\text{LP})$

- assume variable bounds are tight
- $M := \left(\frac{u_i+\ell_i}{2}, \frac{u_j+\ell_j}{2}\right) \in P$
- every facet of P separates at most one of the 4 corners
- optimize along directions from M to each corner

$\Rightarrow$ $P' \supseteq P$ described by at most
  - 4 nontrivial inequalities
  - 4 axis-parallel inequalities

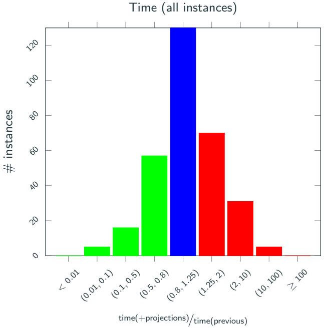


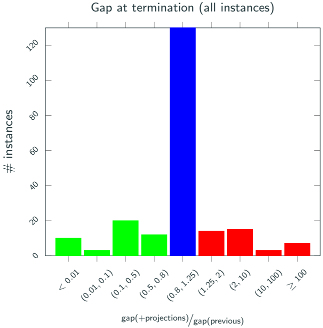
- close connections to optimization-based bound tightening (project LP onto one variable) [Gleixner and Weltge, 2013]
- projections also used to improve bound tightening on $x_i x_j$

[5] Details: Benjamin Müller, Felipe Serrano, Ambros Gleixner, Using two-dimensional Projections for Stronger Separation and Propagation of Bilinear Terms, 2019, ZIB-Report 19-15

Impact of computing and utilizing 2D projections

	previous	+ projections
# solved (out of 1618)	857	857
# solved by both	849	
# solved by both and affected	254	
mean time (on solved&affected)	16.4s	17.3s





Linearizations of products of binary variables

Linearize

$$\prod_{i=1}^{n} x_i, \qquad x_i \in \{0,1\}:$$

- replace by a new variable $z \in \{0,1\}$
- if $n = 2$, add linear constraints $z \leq x_1$, $z \leq x_2$, $z \geq x_1 + x_2 - 1$
- if $n > 2$, add "and"-constraint $z = \bigwedge_{i=1}^{n} x_i$ (specialized constraint handler)

Linearizations of products of binary variables

Linearize

$$\prod_{i=1}^{n} x_i, \qquad x_i \in \{0,1\} :$$

- replace by a new variable $z \in \{0,1\}$
- if $n = 2$, add linear constraints $z \leq x_1$, $z \leq x_2$, $z \geq x_1 + x_2 - 1$
- if $n > 2$, add "and"-constraint $z = \bigwedge_{i=1}^{n} x_i$ (specialized constraint handler)

Linearize

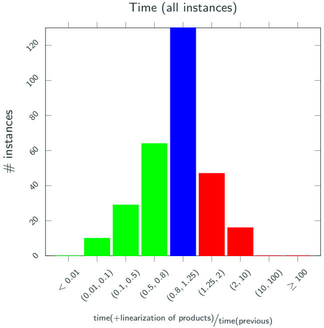
$$y \sum_{j=1}^{n} a_j x_j, \qquad x_j \in \{0,1\}, \qquad n \geq 50 :$$

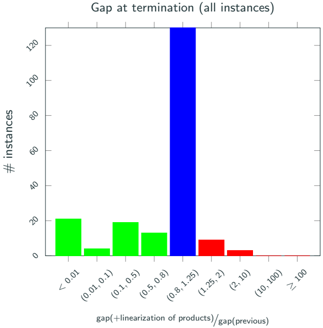
- replace by a new variable $z \in \{0,1\}$, and
- add linear constraints

$$M^L y \leq z \leq M^U y,$$
$$\sum_j a_j x_j - M^U(1-y) \leq z \leq \sum_j a_j x_j - M^L(1-y)$$

Impact of linearization of products of binary variables

	previous	+ linearization
# solved (out of 1618)	857	879
# solved by both	857	
# solved by both and affected	70	
mean time (on solved&affected)	24.3s	16.9s





Detecting of convexity

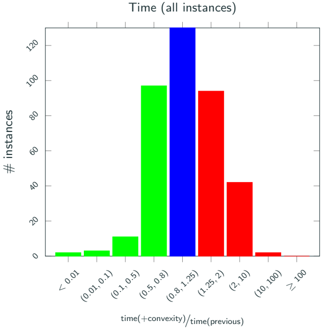
- analyze expressions using a set of rules, e.g.,

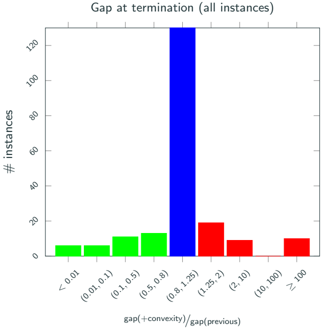
$$
\begin{aligned}
f(x) \text{ convex} &\Rightarrow\ a \cdot f(x) \begin{cases} \text{convex}, & a \geq 0 \\ \text{concave}, & a \leq 0 \end{cases} \\
f(x), g(x) \text{ convex} &\Rightarrow\ f(x) + g(x) \text{ convex} \\
f(x) \text{ concave} &\Rightarrow\ \log(f(x)) \text{ concave} \\
f(x) = \prod_i x_i^{e_i}, x_i \geq 0 &\Rightarrow\ f(x) \begin{cases} \text{convex}, & e_i \leq 0\ \forall i \\ \text{convex}, & \exists j : e_i \leq 0\ \forall i \neq j;\ \sum_i e_i \geq 1 \\ \text{concave}, & e_i \geq 0\ \forall i;\ \sum_i e_i \leq 1 \end{cases}
\end{aligned}
$$

- find maximal convex subexpressions
- underestimate via gradient-cuts

Impact of convexity detection

	previous	+ convexity
# solved (out of 1618)	879	875
# solved by both	868	
# solved by both and affected	325	
mean time (on solved&affected)	14.3s	14.7s





“On/off“-terms

Given $f(x)$ convex with x semicontinuous, i.e., there exists binary variable y such that

$$\begin{aligned} x &= x_0, \quad \text{if } y = 0, \\ x &\in [\ell, u], \quad \text{if } y = 1. \end{aligned}$$

“On/off“-terms

Given $f(x)$ convex with x semicontinuous, i.e., there exists binary variable y such that

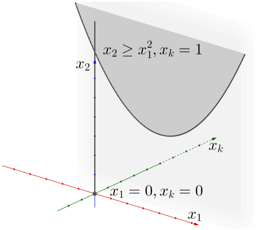
$$x = x_0, \quad \text{if } y = 0,$$
$$x \in [\ell, u], \quad \text{if } y = 1.$$

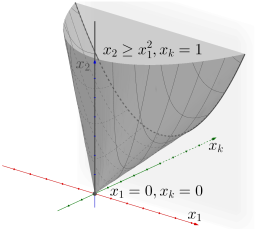
For $x_0 = 0$, $f(0) = 0$, the perspective cut [Frangioni, Gentile, 2006]

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

is valid for the disjunctive set

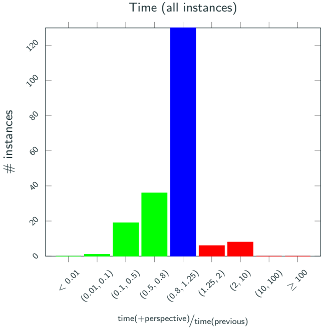
$$\{(x, y, z) : x = x_0, y = 0, f(x_0) \leq w\} \cup \{(x, y, z) : x \in [\ell, u], y = 1, f(x) \leq w\}.$$

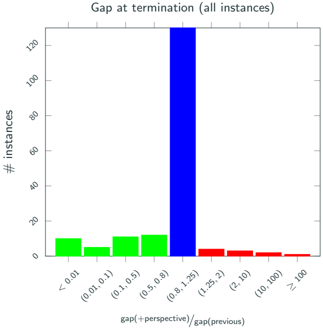




Impact of perspective cuts

	previous	+ perspective
# solved (out of 1618)	875	883
# solved by both	874	
# solved by both and affected	112	
mean time (on solved&affected)	19.8s	15.8s





Symmetry detection

Example:

$$\begin{aligned} \max\ & x_1 + x_2 + x_3 \\ \text{s.t. } & x_1 + x_2 \geq 2 \\ & \sqrt{x_1^2 + x_2^2 + x_3^2} \leq 5 \end{aligned}$$

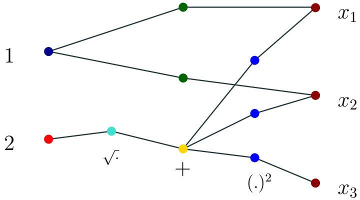
Observation: For any feasible solution, exchanging x_1 and x_2 provides a new feasible solution with same objective value.

Example:

$$\max x_1 + x_2 + x_3$$
$$\text{s.t. } x_1 + x_2 \geq 2$$
$$\sqrt{x_1^2 + x_2^2 + x_3^2} \leq 5$$

Observation: For any feasible solution, exchanging x_1 and x_2 provides a new feasible solution with same objective value.

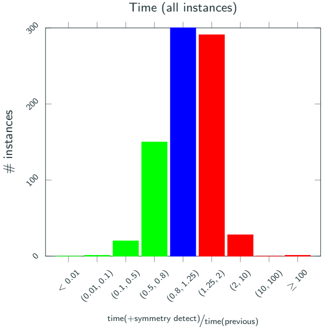
- can be detected by finding automorphisms on a vertex-colored graph [Liberti 2010]

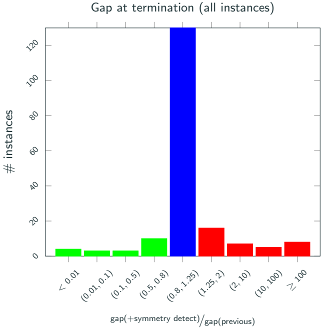


- SCIP aims to find and break symmetries on binary variables [SCIP 5 report, 2017]

Impact of symmetry

	previous	+ symmetry detect
# solved (out of 1618)	883	881
# solved by both	875	
# solved by both and affected	58	
mean time (on solved&affected)	27.1s	19.1s





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

Conclusion

Summary

The handling of nonlinear constraints in SCIP is rewritten.

The new code will be nicer, better, faster, greater:

- less issues with slightly infeasible solutions
- easier to extend by own operators and structure-exploiting algorithms

Core: new constraint handler (`cons_expr`)

Summary

The handling of nonlinear constraints in SCIP is rewritten.

The new code will be nicer, better, faster, greater:

- less issues with slightly infeasible solutions
- easier to extend by own operators and structure-exploiting algorithms

Core: new constraint handler (`cons_expr`)

Expression Handler:

- var, value, sum, product, pow, signed pow, abs, exp, log, cos, sin, entropy

Summary

The handling of nonlinear constraints in SCIP is rewritten.

The new code will be nicer, better, faster, greater:

- less issues with slightly infeasible solutions
- easier to extend by own operators and structure-exploiting algorithms

Core: new constraint handler (`cons_expr`)

Expression Handler:

- var, value, sum, product, pow, signed pow, abs, exp, log, cos, sin, entropy

Nonlinearity Handler:

- quadratic: recognize and separate convex quadratic; domain propagation
- bilinear: tighter estimators and bounds for $x_i x_j$ over polytope
- convex: recognize some simple general convexities, separate by linearization
- perspective: perspective estimators for convex functions in semicontinuous vars.
- (default: wrap around expression handler)

Summary

The handling of nonlinear constraints in SCIP is rewritten.

The new code will be nicer, better, faster, greater:

- less issues with slightly infeasible solutions
- easier to extend by own operators and structure-exploiting algorithms

Core: new constraint handler (`cons_expr`)

Expression Handler:

- var, value, sum, product, pow, signed pow, abs, exp, log, cos, sin, entropy

Nonlinearity Handler:

- quadratic: recognize and separate convex quadratic; domain propagation
- bilinear: tighter estimators and bounds for $x_i x_j$ over polytope
- convex: recognize some simple general convexities, separate by linearization
- perspective: perspective estimators for convex functions in semicontinuous vars.
- (default: wrap around expression handler)

New separator: RLT

Summary

The handling of nonlinear constraints in SCIP is rewritten.

The new code will be nicer, better, faster, greater:

- less issues with slightly infeasible solutions
- easier to extend by own operators and structure-exploiting algorithms

Core: new constraint handler (`cons_expr`)

Expression Handler:

- var, value, sum, product, pow, signed pow, abs, exp, log, cos, sin, entropy

Nonlinearity Handler:

- quadratic: recognize and separate convex quadratic; domain propagation
- bilinear: tighter estimators and bounds for $x_i x_j$ over polytope
- convex: recognize some simple general convexities, separate by linearization
- perspective: perspective estimators for convex functions in semicontinuous vars.
- (default: wrap around expression handler)

New separator: RLT

Symmetry detection

Not ready yet, but getting closer

	classic code	new code
# solved (out of 1618)	827	881
# solved by both	748	
mean time (on solved by both)	4.26s	5.19s