(MI)NLPLib 2

Stefan Vigerske

GAMS

16th July 2015

ISMP, Pittsburgh
## Model instance collections

Collecting optimization problems has been a popular “hobby” for long time, e.g.,

<table>
<thead>
<tr>
<th>first release</th>
<th>library</th>
<th>problem types</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>Netlib</td>
<td>Linear Programming</td>
</tr>
<tr>
<td>1992</td>
<td>MIPLIB</td>
<td>Mixed-Integer Programming</td>
</tr>
<tr>
<td>1993</td>
<td>CUTE</td>
<td>Nonlinear Programming</td>
</tr>
<tr>
<td>1998</td>
<td>SDPLib</td>
<td>Semidefinite Programming</td>
</tr>
<tr>
<td>1999</td>
<td>CSPLib</td>
<td>Constraint Satisfaction Programming</td>
</tr>
<tr>
<td>199x</td>
<td>MacMINLP</td>
<td>Mixed-Integer Nonlinear Programming</td>
</tr>
<tr>
<td>2001</td>
<td>GAMS World</td>
<td>LP, MIP, NLP, MINLP, SOCP, MPEC</td>
</tr>
<tr>
<td>2003</td>
<td>COCONUT</td>
<td>Nonlinear and Constraint Satisfaction Programming</td>
</tr>
<tr>
<td>2008</td>
<td>mintOC</td>
<td>Mixed-Integer Optimal Control</td>
</tr>
<tr>
<td>2009</td>
<td>minlp.org</td>
<td>MINLP, General Disjunctive Programming</td>
</tr>
<tr>
<td>2011</td>
<td>POLIP</td>
<td>Mixed-Integer Polynomial Programming</td>
</tr>
<tr>
<td>2014</td>
<td>CBLIB</td>
<td>Conic Programming</td>
</tr>
</tbody>
</table>

- for solver developers, access to a wide set of interesting problem instances with different characteristics has always been important
- commercial solver vendors test their solver on thousands of test problems before releasing a new software version
- the evaluation of algorithmic improvements (w.r.t. robustness and efficiency) requires well-balanced test sets of significantly many real-world instances
MINLPLib and GLOBALLib

- Initiated in 2001 (as part of GamsWorld/MinlpWorld/GlobalWorld): M. Bussieck, A. Drud, and A. Meeraus
- MINLPLib – A Collection of Test Models for Mixed-Integer Nonlinear Programming
- “white-box” NLPs (GLOBALLib) and MINLPs (MINLPLib)

- frequently used for testing, but also benchmarking
MINLPLib and GLOBALLib Instances

- scalar GAMS format

```
Variables  x1,x2,b3,b4,b5,objvar;
Positive Variables x1,x2;
Binary Variables b3,b4,b5;
Equations   e1,e2,e3,e4,e5,e6;
e1..   - 2*x1 - 3*x2 - 1.5*b3 - 2*b4 + 0.5*b5 + objvar =E= 0;
e2..   sqr(x1) + b3 =E= 1.25;
e3..   x2**1.5 + 1.5*b4 =E= 3;
e4..   x1 + b3 =L= 1.6;
e5..   1.333*x2 + b4 =L= 3;
e6..   - b3 - b4 + b5 =L= 0;
```
MINLPLib and GLOBALLib Instances

- scalar GAMS format

Variables  x1,x2,b3,b4,b5,objvar;
Positive Variables x1,x2;
Binary Variables b3,b4,b5;
Equations  e1,e2,e3,e4,e5,e6;
e1..  - 2*x1 - 3*x2 - 1.5*b3 - 2*b4 + 0.5*b5 + objvar =E= 0;
e2..  sqr(x1) + b3 =E= 1.25;
e3..  x2**1.5 + 1.5*b4 =E= 3;
e4..  x1 + b3 =L= 1.6;
e5..  1.333*x2 + b4 =L= 3;
e6..  - b3 - b4 + b5 =L= 0;

- varying from small scale (great for debugging!) to large scale real world instances (agricultural economics, chemical-, civil-, and electrical engineering, finance, management, OR)

- intentionally including instances from badly formulated models or different formulations of the same problem
MINLPLib and GLOBALLib Instances

- **scalar GAMS format**

  Variables  x1, x2, b3, b4, b5, objvar;
  Positive Variables x1, x2;
  Binary Variables b3, b4, b5;
  Equations  e1, e2, e3, e4, e5, e6;
  e1.. - 2*x1 - 3*x2 - 1.5*b3 - 2*b4 + 0.5*b5 + objvar =E= 0;
  e2.. sqr(x1) + b3 =E= 1.25;
  e3.. x2**1.5 + 1.5*b4 =E= 3;
  e4..  x1 + b3 =L= 1.6;
  e5..  1.333*x2 + b4 =L= 3;
  e6.. - b3 - b4 + b5 =L= 0;

- **varying from small scale** (great for debugging!) to **large scale real world** instances (agricultural economics, chemical-, civil-, and electrical engineering, finance, management, OR)

- intentionally including instances from badly formulated models or different formulations of the same problem

- **including solution points** for many instances
MINLPLib and GLOBALLib Instances

▶ scalar GAMS format

Variables x1,x2,b3,b4,b5, objvar;
Positive Variables x1,x2;
Binary Variables b3,b4,b5;
Equations e1,e2,e3,e4,e5,e6;
e1.. - 2*x1 - 3*x2 - 1.5*b3 - 2*b4 + 0.5*b5 + objvar =E= 0;
e2.. sqr(x1) + b3 =E= 1.25;
e3.. x2**1.5 + 1.5*b4 =E= 3;
e4.. x1 + b3 =L= 1.6;
e5.. 1.333*x2 + b4 =L= 3;
e6.. - b3 - b4 + b5 =L= 0;

▶ varying from small scale (great for debugging!) to large scale real world instances (agricultural economics, chemical-, civil-, and electrical engineering, finance, management, OR)

▶ intentionally including instances from badly formulated models or different formulations of the same problem

▶ including solution points for many instances

▶ solely an instance collection, i.e., consisting of instantiations of models by specific data sets
MINLPLib and GLOBALLib History

- instances were harvested from existing collections, initially:
  - GAMS Model Library
  - MacMINLP (Leyffer)
  - MINOPT library (Floudas)
  - Handbook of Test Problems in Local and Global Optimization (Floudas et.al.)
- 2001 – 2009: maintained by Michael Bussieck
- new instances were added
- new incumbent solutions were added
- in 2009: Michael “volunteered” me as maintainer
MINLPLib 2

MINLPLib Instance Listing

Task:

- Adding new problem instances:
  - both convex and nonconvex problems
  - (MI)QPs, (MI)QCQPs, and (MI)NLPs
  - easy solvable, solvable, difficult to solve, but not trivial

- Categorizing instances
  - convexity
  - problem type (quadratic, polynomial, general nonlinear)
  - function types (powers, exp/log, trigonometric, ...)
  - solved to global optimality?

- Providing feasible best known solutions

Work in progress, current version publicly available:
New NLP and MINLP Instances

MINLPLib 2 - Number of Instances

- GLOBALLib + MINLPLib 1
- MINLPLib 2
Sources of newly added instances

Harvesting mainly from

- CMU-IBM open source MINLP project (convex MINLPs)
- minlp.org
- POLIP (polynomial MINLPs)
## Instance Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>#instances</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMS</td>
<td>.gms</td>
<td><strong>1363</strong> (no Gamma, latest additions missing)</td>
</tr>
<tr>
<td>AIMMS</td>
<td>.ams</td>
<td>1352 (no Gamma, latest additions missing)</td>
</tr>
<tr>
<td>AMPL</td>
<td>.mod</td>
<td>1337 (no erroff/signpower/Gamma/...)</td>
</tr>
<tr>
<td>AMPL</td>
<td>.nl</td>
<td>1331 (no erroff/signpower/Gamma/..., latest missing)</td>
</tr>
<tr>
<td>OSIL</td>
<td>XML</td>
<td>1342 (no signpower/Gamma/...)</td>
</tr>
<tr>
<td>CPLEX LP</td>
<td>.lp</td>
<td>667 (limited to quadratics)</td>
</tr>
<tr>
<td>PIP</td>
<td>.pip</td>
<td>770 (limited to polynomial)</td>
</tr>
</tbody>
</table>
Problem types

Equation Types

GLOBALLib + MINLPLib 1
new in MINLPLib 2

Operands in general nonlinear functions

mul
div
sqr
log
exp
vcpower
sqrt
sin
power
cos
cvpower
log10
errorf
abs
rpower
mod
min
signpower
centropy
gamma
Sparsity Pattern – Examples

Radiation Therapy  Feed Tray Location  Asset Management  Quadratic Assignment

dosemin2d  eg_all_s  feedtray2  johnall  mbtd  qapw

(top: Objective Gradient and Jacobian; bottom: Lagrangian Hessian)
Sparsity Pattern – Examples (cont.)

Jacobian densitymod (Density modification based on single-crystal X-ray diffraction data; 23529 vars, 550 cons.)

Jacobian lop97ic (Rail Line Optimization, MIQCQP)

milinfract (Solving Mixed-Integer Linear Fractional Programming Problems with Dinkelbach’s Algorithm)

Objective Gradient + Jacobian

Lagrangian Hessian
(Non)Convexity Detection for Functions

Analyze the Hessian:

- Given twice differentiable function $h(x)$ and variable bounds $[\underline{x}, \bar{x}]$.
- Compute the spectrum of the Hessian in one random point and conclude
  - convexity/concavity/indefiniteness if $h(x)$ is quadratic
  - nonconvexity/nonconcavity if $h(x)$ is general nonlinear
(Non)Convexity Detection for Functions

Analyze the Hessian:
- Given twice differentiable function $h(x)$ and variable bounds $[\underline{x}, \bar{x}]$.
- Compute the spectrum of the Hessian in one random point and conclude
  - convexity/concavity/indefiniteness if $h(x)$ is quadratic
  - nonconvexity/nonconcavity if $h(x)$ is general nonlinear

Analyze the Algebraic Expression:

\[
f(x) \text{ convex } \Rightarrow a \cdot f(x) \begin{cases} \text{convex, } & a \geq 0 \\ \text{concave, } & a \leq 0 \end{cases}
\]

\[
f(x), g(x) \text{ convex } \Rightarrow f(x) + g(x) \text{ convex}
\]

\[
f(x) \text{ concave } \Rightarrow \log(f(x)) \text{ concave}
\]

\[
f(x) = \prod_{i} x_i^{e_i}, x_i \geq 0 \Rightarrow f(x) \begin{cases} \text{convex, } & e_i \leq 0 \forall i \\ \text{convex, } & \exists j : e_i \leq 0 \forall i \neq j; \sum_i e_i \geq 1 \\ \text{concave, } & e_i \geq 0 \forall i; \sum_i e_i \leq 1 \end{cases}
\]
(Non)Convexity Detection for Functions

Analyze the Hessian:

- Given twice differentiable function \( h(x) \) and variable bounds \([x, \bar{x}]\).
- Compute the spectrum of the Hessian in one random point and conclude
  - convexity/concavity/indeterminacy if \( h(x) \) is quadratic
  - nonconvexity/nonconcavity if \( h(x) \) is general nonlinear

Analyze the Algebraic Expression:

\[
f(x) \text{ convex } \Rightarrow a \cdot f(x) \begin{cases} \text{convex, } & a \geq 0 \\ \text{concave, } & a \leq 0 \end{cases}
\]

\[
f(x), g(x) \text{ convex } \Rightarrow f(x) + g(x) \text{ convex}
\]

\[
f(x) \text{ concave } \Rightarrow \log(f(x)) \text{ concave}
\]

\[
f(x) = \prod_{i} x_i^{e_i}, x_i \geq 0 \Rightarrow f(x) \begin{cases} \text{convex, } & e_i \leq 0 \ \forall i \\ \text{convex, } & \exists j : e_i \leq 0 \ \forall i \neq j; \ \sum_i e_i \geq 1 \\ \text{concave, } & e_i \geq 0 \ \forall i; \ \sum_i e_i \leq 1 \end{cases}
\]

Analyze manually.
(Non)Convexity in MINLPLib

- Numerical analysis of (pointwise) Hessians by LAPACK.
- Symbolic analysis of expressions by SCIP.
- Mark additional 71 instances (5%) as convex.

MINLPLib instances convexity

- Convex: 26%
- Nonconvex: 71%
- Undecided: 3%

MINLPLib instances convexity pie chart

14 / 64
Solution Points

MINLPLib instances traditionally come with known feasible solution points.

Feasibility checking:
1. compute maximal (unscaled) violation of constraints, variable bounds, and discreteness restrictions
2. uses GAMS/EXAMINER2

Solution polishing:
1. project onto variable bounds
2. round values for discrete variables to exact integers
3. ensure that semicontinuity/semiintegrality and special-ordered-set constraints are exactly satisfied
4. run CONOPT on MINLP with all binary/integer/semi*/SOS variables fixed, start from updated point, scaling disabled, feasibility tolerance 10^{-9}
Solution Points

MINLPLib instances traditionally come with known feasible solution points. For MINLPLib 2, we added

Feasibility checking:

- compute maximal (unscaled) violation of constraints, variable bounds, and discreteness restrictions
- uses GAMS/EXAMINER2
Solution Points

MINLPLib instances traditionally come with known feasible solution points.

For MINLPLib 2, we added

Feasibility checking:

- compute maximal (unscaled) violation of constraints, variable bounds, and discreteness restrictions
- uses GAMS/EXAMINER2

Solution polishing: For a given point,

1. project onto variable bounds
2. round values for discrete variables to exact integers
3. ensure that semicontinuity/semiintegrality and special-ordered-set constraints are exactly satisfied
4. run CONOPT on MINLP with all binary/integer/semi*/SOS variables fixed, start from updated point, scaling disabled, feasibility tolerance $10^{-9}$
Polished Solution Points

MINLPLib 1 solution points w.r.t. infeasibility

MINLPLib 2 solution points w.r.t. infeasibility

Available in two formats:
- GAMS Data Exchange (GDX)
- ASCII (.sol)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1.11803398874989001754</td>
</tr>
<tr>
<td>x2</td>
<td>1.31037069710444997739</td>
</tr>
<tr>
<td>b4</td>
<td>1.00000000000000000000</td>
</tr>
<tr>
<td>b5</td>
<td>1.00000000000000000000</td>
</tr>
<tr>
<td>objvar</td>
<td>7.66718006881313041134</td>
</tr>
</tbody>
</table>
Available in two formats:

**GAMS Data Exchange (GDX)**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1.11803398874989001754</td>
</tr>
<tr>
<td>x2</td>
<td>1.31037069710444997739</td>
</tr>
<tr>
<td>b4</td>
<td>1.00000000000000000000</td>
</tr>
<tr>
<td>b5</td>
<td>1.00000000000000000000</td>
</tr>
<tr>
<td>objvar</td>
<td>7.66718006881313041134</td>
</tr>
</tbody>
</table>

**ASCII (.sol)**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1.11803398874989001754</td>
</tr>
<tr>
<td>x2</td>
<td>1.31037069710444997739</td>
</tr>
<tr>
<td>b4</td>
<td>1.00000000000000000000</td>
</tr>
<tr>
<td>b5</td>
<td>1.00000000000000000000</td>
</tr>
<tr>
<td>objvar</td>
<td>7.66718006881313041134</td>
</tr>
</tbody>
</table>
Dual Bounds

dual bound = \begin{cases} 
\text{lower bound on optimal value,} & \text{if minimization} \\
\text{upper bound on optimal value,} & \text{if maximization} 
\end{cases}
Dual Bounds

dual bound = \begin{cases} 
\text{lower bound on optimal value,} & \text{if minimization} \\
\text{upper bound on optimal value,} & \text{if maximization} 
\end{cases}

Collected dual bounds from

- solvers for general (MI)NLP
  (ANTIGONE, BARON, Couenne, Lindo, SCIP)

- solvers for convex MINLP on proven convex MINLPs
  (AlphaECP, Bonmin BB, Bonmin Hyb)
Dual Bounds

dual bound = \begin{cases} 
\text{lower bound on optimal value,} & \text{if minimization} \\
\text{upper bound on optimal value,} & \text{if maximization} 
\end{cases}

Collected dual bounds from

- solvers for general (MI)NLP
  (ANTIGONE, BARON, Couenne, Lindo, SCIP)
- solvers for convex MINLP on proven convex MINLPs
  (AlphaECP, Bonmin BB, Bonmin Hyb)

But: No way to verify correctness of bound!
Dual Bounds

dual bound = \begin{cases} 
\text{lower bound on optimal value,} & \text{if minimization} \\
\text{upper bound on optimal value,} & \text{if maximization}
\end{cases}

Collected dual bounds from

- solvers for general (MI)NLP (ANTIGONE, BARON, Couenne, Lindo, SCIP)
- solvers for convex MINLP on proven convex MINLPs (AlphaECP, Bonmin BB, Bonmin Hyb)

But: No way to verify correctness of bound!

Conservative approach: Only trust a solvers dual bound claim if it has been verified by at least 2 other solvers.
“Open” instances

Feasible solution points $\oplus$ trusted dual bounds $\Rightarrow$ trusted gap

MINLPLib histogram w.r.t. trusted gap

- GLOBALLib + MINLPLib 1
- new in MINLPLib 2

Number of instances

Trusted Gap

0.0 $\leq 10^{-9}$

1.0 $\geq 1.0$
Query the MINLPLib

Simple script to select instances by specific criteria, e.g.:

- all large convex instances, show # var. and # cons.:
  
  ⬢ $ ./query.py "(nvars > 4242) & (convex == True)" -c nvars -c ncons

<table>
<thead>
<tr>
<th>Instance</th>
<th>nvars</th>
<th>ncons</th>
</tr>
</thead>
<tbody>
<tr>
<td>jbearing100</td>
<td>5304</td>
<td>0</td>
</tr>
<tr>
<td>squfl030-150</td>
<td>4530</td>
<td>4650</td>
</tr>
<tr>
<td>watercontamination0202</td>
<td>106711</td>
<td>107209</td>
</tr>
<tr>
<td>watercontamination0303</td>
<td>107222</td>
<td>108217</td>
</tr>
</tbody>
</table>

- all quadratic instances:
  
  ⬢ ./query.py "npolynomfunc == 0 & nsignomfunc == 0 & ngennlfunc == 0"

- all instances with trigonometric functions:
  
  ⬢ ./query.py "(opsin == True) or (opcos == True)"

- all separable instances, sorted by problem type:
  
  ⬢ ./query.py "nlaghessiannz == nlaghessiandiagnz" -s probtype -c probtype

- all unsolved instances (w.r.t. “trusted” dual bounds), zipped up:
  
  ⬢ ./query.py "gap > 0.1" -c gap -z open.zip
What to do with all these instances?
What to do with all these instances?

<table>
<thead>
<tr>
<th>date</th>
<th>GAMS</th>
<th>ANTIQUE</th>
<th>BARON</th>
<th>COUENNE</th>
<th>LINDO</th>
<th>SCIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/15</td>
<td>24.5α</td>
<td>1.1</td>
<td>15.6.5</td>
<td>0.5</td>
<td>9.0.1983.157</td>
<td>3.2.0</td>
</tr>
</tbody>
</table>
Today: Go Columnwise

<table>
<thead>
<tr>
<th>date</th>
<th>GAMS</th>
<th>ANTIGONE</th>
<th>BARON</th>
<th>COUENNE</th>
<th>LINDO</th>
<th>SCIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>08/11</td>
<td>23.7.3</td>
<td>–</td>
<td>9.3.1</td>
<td>0.3</td>
<td>6.1.1.588</td>
<td>–</td>
</tr>
<tr>
<td>04/12</td>
<td>23.8.2</td>
<td>–</td>
<td>10.2.0</td>
<td>0.4</td>
<td>7.0.1.421</td>
<td>2.1.1</td>
</tr>
<tr>
<td>11/12</td>
<td>23.9.5</td>
<td>–</td>
<td>11.5.2</td>
<td>0.4</td>
<td>7.0.1.497</td>
<td>2.1.2</td>
</tr>
<tr>
<td>02/13</td>
<td>24.0.2</td>
<td>–</td>
<td>11.9.1</td>
<td>0.4</td>
<td>7.0.1.497</td>
<td>3.0</td>
</tr>
<tr>
<td>07/13</td>
<td>24.1.3</td>
<td>1.1</td>
<td>12.3.3</td>
<td>0.4</td>
<td>8.0.1283.385</td>
<td>3.0</td>
</tr>
<tr>
<td>05/14</td>
<td>24.2.3</td>
<td>1.1</td>
<td>12.7.7</td>
<td>0.4</td>
<td>8.0.1694.498</td>
<td>3.0</td>
</tr>
<tr>
<td>09/14</td>
<td>24.3.3</td>
<td>1.1</td>
<td>14.0.3</td>
<td>0.4</td>
<td>8.0.1694.550</td>
<td>3.1</td>
</tr>
<tr>
<td>06/15</td>
<td>24.4.6</td>
<td>1.1</td>
<td>14.4.0</td>
<td>0.4</td>
<td>9.0.1983.157</td>
<td>3.1</td>
</tr>
<tr>
<td>07/15</td>
<td>24.5α</td>
<td>1.1</td>
<td>15.6.5</td>
<td>0.5</td>
<td>9.0.1983.157</td>
<td>3.2</td>
</tr>
</tbody>
</table>

- **ANTIGONE** by R. Misener (Imperial College) and Ch. Floudas (Texas A&M)
- **BARON** by N. Sahinidis (CMU), M. Tawarmalani (Purdue), et.al.
- **Couenne** by P. Belotti (now FICO), et.al.; open-source (COIN-OR)
- **Lindo API** by Lindo Systems Inc.
- **SCIP** by Zuse Institute Berlin, TU Darmstadt, FAU Erlangen; free for academic use

Quantify Improvements of global MINLP solvers over the last 4 years!
Which instances to run?

Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?
Which instances to run?

Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?

✓ large number of instances
  ⇒ 40 solver versions × 1363 instances = 54520 runs (!)

✓ wide variety of applications
   × dominance of certain models, e.g.,
     ▶ 32 block layout design problems
     ▶ 60 small investor portfolio optimization instances
     ▶ ...

   × many trivial, some hopeless, some numerically dubious instances

Thus, need to select a reasonable subset of (e.g., 87) instances.

With 15 co-authors and 8 months of time, this would be no problem.

But with 3 weeks until ISMP:
Apply the P.I.T.T. heuristic.
Which instances to run?

Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?

✓ large number of instances
  ⇒ 40 solver versions × 1363 instances = 54520 runs (!)
✓ wide variety of applications
Which instances to run?

Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?

✓ large number of instances
  ⇒ 40 solver versions × 1363 instances = 54520 runs (!)

✓ wide variety of applications

✗ dominance of certain models, e.g.,
  ▶ 32 block layout design problems
  ▶ 60 small investor portfolio optimization instances
  ▶ ...

Thus, need to select a reasonable subset of (e.g., 87) instances.

With 15 co-authors and 8 months of time, this would be no problem.

But with 3 weeks until ISMP: Apply the P.I.T.T. heuristic.
Which instances to run?
Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?

✓ large number of instances
  ⇒ 40 solver versions × 1363 instances = 54520 runs (!)

✓ wide variety of applications

✗ dominance of certain models, e.g.,
  ▶ 32 block layout design problems
  ▶ 60 small investor portfolio optimization instances
  ▶ ...

✗ many trivial, some hopeless, some numerically dubious instances
Which instances to run?

Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?

✓ large number of instances
  ⇒ 40 solver versions × 1363 instances = 54520 runs (!)

✓ wide variety of applications

× dominance of certain models, e.g.,
  ▶ 32 block layout design problems
  ▶ 60 small investor portfolio optimization instances
  ▶ ...

× many trivial, some hopeless, some numerically dubious instances

Thus, need to select a reasonable subset of (e.g., 87) instances.
Which instances to run?
Does the current MINLPLib 2 with its 1363 instances qualify as a good test set?

✓ large number of instances
   ⇒ 40 solver versions × 1363 instances = 54520 runs (!)
✓ wide variety of applications
✗ dominance of certain models, e.g.,
   ▶ 32 block layout design problems
   ▶ 60 small investor portfolio optimization instances
   ▶ ...
✗ many trivial, some hopeless, some numerically dubious instances

Thus, need to select a reasonable subset of (e.g., 87) instances.

With 15 co-authors and 8 months of time, this would be no problem.

But with 3 weeks until ISMP: Apply the P.I.T.T. heuristic.
Prune Instances by Tractability and Triviality Heuristic

1. Remove intractable instances
   ▶ consider only the 881 instances that are marked as solved in MINLPLib 2

![Graph showing the distribution of instances based on number of discrete variables and nonlinear nonzeros in Jacobian](image)

![Histogram showing instances by trusted gap in MINLPLib](image)
Prune Instances by Tractability and Triviality Heuristic

2. For each solver separately:
   ▶ Remove instances that are solved within 60 seconds by the oldest solver version (e.g., as in GAMS 23.7).
   ▶ Remove instances that the solver cannot handle (due to trigonometric functions, SOS, ...).

In case of SCIP:
Prune Instances by Tractability and Triviality Heuristic

For SCIP, this leaves 312 instances:

<table>
<thead>
<tr>
<th>alklylation</th>
<th>clay0304h</th>
<th>ex6_1_1</th>
<th>fo9_ar5_1</th>
<th>house</th>
</tr>
</thead>
<tbody>
<tr>
<td>arki0003</td>
<td>clay0305h</td>
<td>ex6_1_3</td>
<td>gasnet</td>
<td>jbearing25</td>
</tr>
<tr>
<td>arki0005</td>
<td>crudeoil_lee2_10</td>
<td>ex6_2_12</td>
<td>genpooling_meyer04</td>
<td>jbearing75</td>
</tr>
<tr>
<td>arki0006</td>
<td>crudeoil_lee3_07</td>
<td>ex6_2_14</td>
<td>ghg_1veh</td>
<td>johnall</td>
</tr>
<tr>
<td>arki0019</td>
<td>crudeoil_lee3_08</td>
<td>ex6_2_8</td>
<td>ghg_3veh</td>
<td>kall_circles_c6a</td>
</tr>
<tr>
<td>arki0024</td>
<td>crudeoil_lee3_09</td>
<td>ex6_2_9</td>
<td>glider100</td>
<td>kall_circles_c6b</td>
</tr>
<tr>
<td>autocorr_ber20-10</td>
<td>crudeoil_lee3_10</td>
<td>ex7_2_4</td>
<td>graphpart_2g-0066-0066</td>
<td>kall_circles_c7a</td>
</tr>
<tr>
<td>autocorr_ber20-15</td>
<td>crudeoil_li06</td>
<td>ex8_1_7</td>
<td>graphpart_2g-0077-0077</td>
<td>kall_circles_c8a</td>
</tr>
<tr>
<td>autocorr_ber25-06</td>
<td>csched1a</td>
<td>ex8_2_1b</td>
<td>graphpart_2g-0088-0088</td>
<td>kall_circlespolygons_c1p1</td>
</tr>
<tr>
<td>autocorr_ber25-13</td>
<td>edgecross10-060</td>
<td>ex8_2_4b</td>
<td>graphpart_2g-0099-9211</td>
<td>kall_circlespolygons_c1p1</td>
</tr>
<tr>
<td>autocorr_ber30-04</td>
<td>edgecross10-070</td>
<td>ex8_4_1</td>
<td>graphpart_2pm-0066-0066</td>
<td>kall_circlesrectangles_c3</td>
</tr>
<tr>
<td>autocorr_ber35-04</td>
<td>edgecross10-080</td>
<td>ex8_4_3</td>
<td>graphpart_2pm-0077-0077</td>
<td>kall_circlesrectangles_c3</td>
</tr>
<tr>
<td>batch0812_nc</td>
<td>edgecross14-039</td>
<td>ex8_4_4</td>
<td>graphpart_2pm-0088-0888</td>
<td>kall_congruentcircles_c7</td>
</tr>
<tr>
<td>batches201210m</td>
<td>edgecross14-058</td>
<td>ex8_4_5</td>
<td>graphpart_2pm-0099-0999</td>
<td>kall_diffcircles_10</td>
</tr>
<tr>
<td>bayes2_50</td>
<td>edgecross14-078</td>
<td>ex8_4_8_bnd</td>
<td>graphpart_3g-0334-0334</td>
<td>kall_diffcircles_5b</td>
</tr>
<tr>
<td>blend480</td>
<td>edgecross14-176</td>
<td>filter</td>
<td>graphpart_3g-0344-0344</td>
<td>kall_diffcircles_7</td>
</tr>
<tr>
<td>blend531</td>
<td>edgecross20-040</td>
<td>fin2bb</td>
<td>graphpart_3g-0444-0444</td>
<td>kall_diffcircles_8</td>
</tr>
<tr>
<td>blend718</td>
<td>edgecross22-048</td>
<td>flay05h</td>
<td>graphpart_3pm-0244-0244</td>
<td>kall_diffcircles_9</td>
</tr>
<tr>
<td>blend852</td>
<td>emf1050_5_5</td>
<td>fo7</td>
<td>graphpart_3pm-0333-0333</td>
<td>launch</td>
</tr>
<tr>
<td>carton7</td>
<td>emf1100_5_5</td>
<td>fo8</td>
<td>graphpart_3pm-0334-0334</td>
<td>lop97icx</td>
</tr>
<tr>
<td>casctanks</td>
<td>ethanolh</td>
<td>fo8_ar25_1</td>
<td>graphpart_3pm-0344-0344</td>
<td>mathopt5_7</td>
</tr>
<tr>
<td>cecil_13</td>
<td>ethanolm</td>
<td>fo8_ar2_1</td>
<td>graphpart_3pm-0444-0444</td>
<td>mathopt5_8</td>
</tr>
<tr>
<td>chem</td>
<td>ex1252a</td>
<td>fo9</td>
<td>graphpart_clique-30</td>
<td>mhw4d</td>
</tr>
<tr>
<td>clay0203h</td>
<td>ex14_1_1</td>
<td>fo9_ar25_1</td>
<td>graphpart_clique-40</td>
<td>milinfract</td>
</tr>
<tr>
<td>clay0204h</td>
<td>ex14_1_7</td>
<td>fo9_ar2_1</td>
<td>gsg_0001</td>
<td>minlphix</td>
</tr>
<tr>
<td>clay0205h</td>
<td>ex4_1_5</td>
<td>fo9_ar3_1</td>
<td>hda</td>
<td>minsurf100</td>
</tr>
<tr>
<td>clay0303h</td>
<td>ex4_1_6</td>
<td>fo9_ar4_1</td>
<td>heatexch_trigen</td>
<td>...</td>
</tr>
</tbody>
</table>
Prune Instances by Tractability and Triviality Heuristic

For SCIP, this leaves 312 instances – obvious dominance by some models:

- alkylation
- arki0003
- arki0005
- arki0006
- arki0019
- arki0024
- autocorr_ber20-10
- autocorr_ber20-15
- autocorr_ber25-06
- autocorr_ber25-13
- autocorr_ber30-04
- autocorr_ber35-04
- batch0812_nc
- batchs201210m
- bayes2_50
- blend480
- blend531
- blend718
- blend852
- carton7
- casctanks
- cecil_13
- chem
- clay0203h
- clay0204h
- clay0205h
- clay0303h
- fo9_ar5_1
- gasnet
- genpooling_meyer04
- ghg_1veh
- ghg_3veh
- glider100
- kall_circles_c6a
- kall_circles_c6b
- kall_circles_c7a
- kall_circles_c8a
- kall_circlespolygons_clp
- kall_circlespolygons_clp
- kall_congruentcircles_c7
- kall_congruentcircles_c7
- kall_diffcircles_5b
- kall_diffcircles_7
- kall_diffcircles_8
- kall_diffcircles_9
- kall_diffcircles_10
- lop97icx
- mathopt5_7
- mathopt5_8
- mhw4d
- milinfract
- minlphix
- minsurf100
- heatexch_trigen
- house
- jbearing25
- jbearing75
- johnall
- kall_circles_c6a
- kall_circles_c6b
- kall_circles_c7a
- kall_circles_c8a
- kall_circlespolygons_clp
- kall_circlespolygons_clp
- kall_congruentcircles_c7
- kall_congruentcircles_c7
- kall_diffcircles_5b
- kall_diffcircles_7
- kall_diffcircles_8
- kall_diffcircles_9
- kall_diffcircles_10
- lop97icx
- mathopt5_7
- mathopt5_8
- mhw4d
- milinfract
- minlphix
- minsurf100
- heatexch_trigen
- ...
P.I.T.T.E.D. Heuristic: P.I.T.T. with Eased Dominance

3. Ensure uniqueness of 6-characters-prefix of instances names.
P.I.T.T.E.D. Heuristic: P.I.T.T. with Eased Dominance

3. Ensure uniqueness of 6-characters-prefix of instances names.

alkylation
arki0003
arki0005
arki0006
arki0019
arki0024
autocorr_bern20-10
autocorr_bern20-15
autocorr_bern25-06
autocorr_bern25-13
autocorr_bern30-04
autocorr_bern35-04
batch0812_nc
batchs201210m
bayes2_50
blend480
blend531
blend718
blend852
carton7
casctanks
cecil_13
chem
clay0203h
clay0204h
clay0205h
clay0303h
clay0304h
clay0305h
clay0306h
clay0307h
crudeoil_lee2_10
clay0308h
crudeoil_lee3_07
crudeoil_lee3_08
clay0309h
crudeoil_lee3_09
autocorr_bern20-15
clay0310h
crudeoil_lee3_10
edgecross10-060
edgecross10-070
edgecross10-080
edgecross14-039
edgecross14-058
edgecross14-076
edgecross20-040
edgecross22-048
emfl100_5_5
e6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex8_4_1
filter
emfl050_5_5
e6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd
ex6_1_1
ex6_1_3
ex6_2_12
ex6_2_14
ex6_2_8
ex6_2_9
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_2_4b
ex8_4_1
ex8_4_3
ex8_4_4
ex8_4_5
ex8_4_8_bnd

P.I.T.T.E.D. SCIP testset

In summary:

1. Keep only instances that are marked as solved in MINLPLib 2.
2. Keep only instances that take $\geq 60s$ with oldest version of solver and that can be handled by solver.
3. Reduce instances with similar names.

For SCIP, this reduces from 1363 to 881 to 123 instances:
PITT test set for each solver

Removed easy and unsolvable instances:

- **ANTIGONE**
  - Number of Discrete Variables (+1): 10
  - Number of Nonlinear Nonzeros in Jacobian: 0

- **BARON**
  - Number of Discrete Variables (+1): 10
  - Number of Nonlinear Nonzeros in Jacobian: 0

- **COUENNE**
  - Number of Discrete Variables (+1): 10
  - Number of Nonlinear Nonzeros in Jacobian: 1

- **LINDO**
  - Number of Discrete Variables (+1): 10
  - Number of Nonlinear Nonzeros in Jacobian: 1

- **SCIP**
  - Number of Discrete Variables (+1): 10
  - Number of Nonlinear Nonzeros in Jacobian: 2
PITTED test set for each solver

Removed easy and unsolvable instances, then filter by name:

- **ANTIGONE**
- **BARON**
- **COUENNE**
- **LINDO**
- **SCIP**
Run jobs

<table>
<thead>
<tr>
<th>date</th>
<th>GAMS</th>
<th>ANTIGONE</th>
<th>BARON</th>
<th>COUENNE</th>
<th>LINDO</th>
<th>SCIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>08/11</td>
<td>23.7</td>
<td>–</td>
<td>9.1</td>
<td>0.3</td>
<td>6.1.1.588</td>
<td>–</td>
</tr>
<tr>
<td>04/12</td>
<td>23.8</td>
<td>–</td>
<td>10.2</td>
<td>0.4</td>
<td>7.0.1.421</td>
<td>2.1.1</td>
</tr>
<tr>
<td>11/12</td>
<td>23.9</td>
<td>–</td>
<td>11.5</td>
<td>0.4</td>
<td>7.0.1.497</td>
<td>2.1.2</td>
</tr>
<tr>
<td>02/13</td>
<td>24.0</td>
<td>–</td>
<td>11.9</td>
<td>0.4</td>
<td>7.0.1.497</td>
<td>3.0</td>
</tr>
<tr>
<td>07/13</td>
<td>24.1</td>
<td>1.1</td>
<td>12.3</td>
<td>0.4</td>
<td>8.0.1283.385</td>
<td>3.0</td>
</tr>
<tr>
<td>05/14</td>
<td>24.2</td>
<td>1.1</td>
<td>12.7</td>
<td>0.4</td>
<td>8.0.1694.498</td>
<td>3.0</td>
</tr>
<tr>
<td>09/14</td>
<td>24.3</td>
<td>1.1</td>
<td>14.0</td>
<td>0.4</td>
<td>8.0.1694.550</td>
<td>3.1</td>
</tr>
<tr>
<td>06/15</td>
<td>24.4</td>
<td>1.1</td>
<td>14.4</td>
<td>0.4</td>
<td>9.0.1983.157</td>
<td>3.1</td>
</tr>
<tr>
<td>07/15</td>
<td>24.5α</td>
<td>1.1</td>
<td>15.6</td>
<td>0.5</td>
<td>9.0.1983.157</td>
<td>3.2</td>
</tr>
</tbody>
</table>

for GAMS in $GAMSS ; do
  for SOLVER in $SOLVERS($GAMS) ; do
    for INSTANCE in $TESTSET($SOLVER) ; do
      sbatch --exclusive --time=0:1800 $GAMS $INSTANCE SOLVER=$SOLVER
    done
  done
done

Hardware: Dell PowerEdge M1000e, 48GB RAM, Intel Xeon X5672@3.2GHz
BARON: Fails

A solver failed, if it

- crashed, or
- reported an infeasible point as feasible (tolerance: $10^{-4}$), or
- reported a suboptimal solution as optimal (tolerance: $10^{-4}$)

![Baron version comparison graph]

<table>
<thead>
<tr>
<th>BARON version</th>
<th>% failed instances (out of 188)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3</td>
<td>4.3</td>
</tr>
<tr>
<td>10.2</td>
<td>4.8</td>
</tr>
<tr>
<td>11.5</td>
<td>3.2</td>
</tr>
<tr>
<td>11.9</td>
<td>4.3</td>
</tr>
<tr>
<td>12.3</td>
<td>5.3</td>
</tr>
<tr>
<td>12.7</td>
<td>1</td>
</tr>
<tr>
<td>14.0</td>
<td>1.6</td>
</tr>
<tr>
<td>14.4</td>
<td>1.1</td>
</tr>
<tr>
<td>15.6</td>
<td>1</td>
</tr>
</tbody>
</table>

- blue: same as previous version
- red: additional to previous version
BARON: Solved

Solved: solver did not fail and reports a relative optimality gap $\leq 10^{-4}$
BARON: Solved – What happened?

From the release notes:

**11.0:** “This version comes with a wealth of new branching, relaxation, convexity exploitation, local search, and range reduction techniques.”

**11.5:** “ Improvements in local search” (dive-and-round heuristic for MINLPs, automatically select and switch back and forth between NLP solvers)

**12.3:** “New relaxations for certain types of quadratic problems”, “Improved integer presolve”, “Incorporation of convex envelopes for certain low-dimensional functions”

**14.0:** “Significant advances in the handling of integer programs.” (integer cutting planes, calls to MIP solvers, hybrid LP/MIP/NLP relaxations)
BARON: Found optimal solution

% found optimal solution (out of 188)

BARON version

- 9.3
- 10.2
- 11.5
- 11.9
- 12.3
- 12.7
- 14.0
- 14.4
- 15.6

Legend:
- same as previous version
- additional to previous version
- only in previous version

Baron version 0 found optimal solution

34 / 64
BARON: Solving time on instances that never failed (163)

Overall speedup: 9.00

12.7: “Automatic setting of many options based on problem characteristics and learning algorithms.“One
BARON: Solving time on instances solved by all vers. (69)

Overall speedup: 3.67
ANTIGONE: Fails

% failed instances (out of 189)

same as previous version
additional to previous version

2.6 / 64
ANTIGONE: Solved

% solved instances (out of 189)

ANTIGONE version

<table>
<thead>
<tr>
<th>Version</th>
<th>61.9</th>
<th>62.4</th>
<th>62.4</th>
<th>59.3</th>
<th>59.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- blue: same as previous version
- red: additional to previous version
- grey: only in previous version
ANTIGONE: Found optimal solution

% found optimal solution (out of 189)

ANTIGONE version

- 83.6
- 85.2
- 84.6
- 80.4

Legend:
- Same as previous version
- Additional to previous version
- Only in previous version
ANTIGONE: Solving time on instances that never failed (177)
ANTIGONE: Solving time on instances solved by all (110)

![Graph showing mean solving time and speedup](image)

- **Mean time**: 1.19, 1.03, 0.99, 1.00
- **Wilcoxon p-value**: 41 / 64
COUENNE: Fails

% failed instances (out of 124)

- same as previous version
- additional to previous version
- only in previous version

COUENNE version

- 0.3: 16.9%
- 0.4: 37.1%
- 0.4: 20.9%
- 0.4: 22.6%
- 0.4: 20.1%
- 0.4: 20.1%
- 0.4: 19.3%
- 0.4: 17.7%
- 0.5: 14.6%
COUENNE: Solved

% solved instances (out of 124)

- blue: same as previous version
- red: additional to previous version
- dotted: only in previous version

COUENNE version

- 0.3
- 0.4
- 0.4
- 0.4
- 0.4
- 0.4
- 0.4
- 0.5

Values:

- 29
- 21.8
- 27.5
- 25
- 25
- 25.8
- 26.6
- 26.6
- 34.7
COUENNE: Found optimal solution

<table>
<thead>
<tr>
<th>COUENNE version</th>
<th>% found optimal solution (out of 124)</th>
<th>same as previous version</th>
<th>additional to previous version</th>
<th>only in previous version</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>46</td>
<td>35.5</td>
<td>4.5</td>
<td>6.0</td>
</tr>
<tr>
<td>0.4</td>
<td>45.2</td>
<td>43.5</td>
<td>1.7</td>
<td>0.0</td>
</tr>
<tr>
<td>0.4</td>
<td>43.5</td>
<td>42.7</td>
<td>0.8</td>
<td>0.0</td>
</tr>
<tr>
<td>0.4</td>
<td>40.3</td>
<td>41.9</td>
<td>1.4</td>
<td>0.0</td>
</tr>
<tr>
<td>0.4</td>
<td>41.9</td>
<td>41.9</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.4</td>
<td>43.6</td>
<td>43.6</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.5</td>
<td>46.0</td>
<td>43.6</td>
<td>2.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>
COUENNE: Solving time on instances that never failed (65)

Overall speedup: 1.35
Dear Couenne users,

this is to announce the 0.4 stable version of Couenne. There are a number of additions and improvements, including:

1) a Feasibility Pump heuristic for non-convex MINLP, developed with Timo Berthold at the ZIB institute.

2) Orbital Branching for MINLP, developed with Jim Ostrowski and Leo Liberti.

3) Fixed Point Bound tightening, a bound reduction procedure developed with Sonia Cafieri, Jon Lee, and Leo Liberti.

4) "semi-auxiliaries", i.e., auxiliary variables defined as \( y \geq f(x) \) or \( y \leq f(x) \) instead of just \( y = f(x) \). The purpose is to save on the number of auxiliaries generated and hence on the size of the LP relaxation.


6) various bug fixes.

Release 0.4.0 is a snapshot of the new stable version. The new features will soon be documented in Couenne's user manual, available at [http://www.coin-or.org/Couenne/couenne-user-manual.pdf](http://www.coin-or.org/Couenne/couenne-user-manual.pdf)

Happy MINLPing,

Pietro

---

Pietro Belotti
Dept. of Mathematical Sciences
Clemson University
email: pbelott at clemson.edu
phone: 864-656-6765
web: myweb.clemson.edu/~pbelott
Dear Couenne users,

this is to announce the 0.4 stable version of Couenne. There are a number of additions and improvements, including:

1) a Feasibility Pump heuristic for non-convex MINLP, developed with Timo Berthold at the ZIB institute.

2) Orbital Branching for MINLP, developed with Jim Ostrowski and Leo Liberti.

3) Fixed Point Bound tightening, a bound reduction procedure developed with Sonia Cafieri, Jon Lee, and Leo Liberti.

4) "semi-auxiliaries", i.e., auxiliary variables defined as $y \geq f(x)$ or $y \leq f(x)$ instead of just $y = f(x)$. The purpose is to save on the number of auxiliaries generated and hence on the size of the LP relaxation.


6) various bug fixes.

Release 0.4.0 is a snapshot of the new stable version. The new features will soon be documented in Couenne’s user manual, available at http://www.coin-or.org/Couenne/couenne-user-manual.pdf

Happy MINLPing,

Pietro

---

Pietro Belotti
Dept. of Mathematical Sciences
Clemson University
email: pbelott at clemson.edu
phone: 864-656-6765
web: myweb.clemson.edu/~pbelott

1. Feasibility Pump
   feasibility_pump no

2. Orbital Branching
   orbital_branching no

3. Fixed Point BT
   fixpoint_bt 0

4. “semi-auxiliaries”
   use_semiaux yes

5. Two-Implied BT
   two_implied_bt 0

6. various bug fixes
COUENNE: Solving time on instances solved by all (16)

Overall speedup: 2.16
LINDO: Fails

% failed instances (out of 128)

LINDO version

- same as previous version
- additional to previous version
- only in previous version

5.5 / 6.1
14.8 / 7.0
24.2 / 7.0
23.4 / 7.0
6.2 / 8.0
8.6 / 8.0
8.6 / 8.0
12.4 / 9.0
12.5 / 9.0
LINDO: Solved

LINDO version | % solved instances (out of 128)
---|---
6.1 | 39.1
7.0 | 35.2
7.0 | 34.4
7.0 | 32.8
8.0 | 48.4
8.0 | 48.5
8.0 | 46.9
9.0 | 43
9.0 | 43

LINDO 8.0: improvements in primal heuristics for MIP (feas. pump) and nonconvex NLP (multistart)
LINDO: Found optimal solution

% found optimal solution (out of 128)

LINDO version

<table>
<thead>
<tr>
<th>Version</th>
<th>Same as Previous</th>
<th>Additional to Previous</th>
<th>Only in Previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>59.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>53.1</td>
<td>47.6</td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>46.1</td>
<td>64.8</td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>63.3</td>
<td>66.4</td>
<td>62.5</td>
</tr>
<tr>
<td>8.0</td>
<td>62.5</td>
<td></td>
<td>62.5</td>
</tr>
<tr>
<td>9.0</td>
<td></td>
<td></td>
<td>62.5</td>
</tr>
<tr>
<td>9.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LINDO: Solving time on instances that never failed (72)

Overall speedup: 5.48
LINDO: Solving time on instances solved by all vers. (16)

Overall speedup: 3.02
SCIP: Fails

% failed instances (out of 123)

- same as previous version
- additional to previous version
- only in previous version

SCIP version

2.1: 13%
2.1: 9%
3.0: 10.5%
3.0: 8.9%
3.0: 8.1%
3.1: 1.6%
3.1: 1.6%
3.2: 1.6%
SCIP: Solved

% solved instances (out of 123)

- same as previous version
- additional to previous version
- only in previous version

SCIP version

- 2.1: 32.5%
- 3.0: 44.7%
- 3.1: 55.3%
- 3.2: 54.5%
SCIP: Found optimal solution

- same as previous version
- additional to previous version
- only in previous version

<table>
<thead>
<tr>
<th>SCIP version</th>
<th>% found optimal solution (out of 123)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>61.8</td>
</tr>
<tr>
<td>2.1</td>
<td>60.2</td>
</tr>
<tr>
<td>3.0</td>
<td>65.1</td>
</tr>
<tr>
<td>3.0</td>
<td>66.6</td>
</tr>
<tr>
<td>3.0</td>
<td>68.3</td>
</tr>
<tr>
<td>3.1</td>
<td>75.6</td>
</tr>
<tr>
<td>3.1</td>
<td>73.2</td>
</tr>
<tr>
<td>3.2</td>
<td>77.2</td>
</tr>
</tbody>
</table>
SCIP: Solving time on instances that never failed (96)

Overall speedup: 4.49
SCIP: Solving time on instances solved by all vers. (31)

Overall speedup: 2.42
“Virtual Best” Solver

- common subset of instances
- for each instance and GAMS version, pick best results among all solvers

![Graphs showing scatter plots of number of discrete variables vs. number of nonlinear nonzeros in Jacobian for different solvers and the virtual best solution.]

<table>
<thead>
<tr>
<th>Solver</th>
<th>Number of Discrete Variables (+1)</th>
<th>Number of Nonlinear Nonzeros in Jacobian</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANTIGONE</td>
<td>![Graph for ANTIGONE]</td>
<td>![Graph for ANTIGONE]</td>
</tr>
<tr>
<td>BARON</td>
<td>![Graph for BARON]</td>
<td>![Graph for BARON]</td>
</tr>
<tr>
<td>COUENNE</td>
<td>![Graph for COUENNE]</td>
<td>![Graph for COUENNE]</td>
</tr>
<tr>
<td>LINDO</td>
<td>![Graph for LINDO]</td>
<td>![Graph for LINDO]</td>
</tr>
<tr>
<td>SCIP</td>
<td>![Graph for SCIP]</td>
<td>![Graph for SCIP]</td>
</tr>
</tbody>
</table>

Virtual Best: 58 / 64
Virtual Best: Fails

% failed instances (out of 71)

GAMS version

- Same as previous version
- Additional to previous version

1.4
Virtual Best: Solved

% solved instances (out of 71)

GAMS version

23.7 23.8 23.9 24.0 24.1 24.2 24.3 24.4 24.5
Virtual Best: Solving time on instances that never failed (70)

Overall speedup: 14.84
Virtual Best: Solving time on instances that never failed (70)

Overall speedup: 14.84
Virtual Best: Solving time on instances solved by all (17)

Overall speedup: 14.30
End.

http://www.gamsworld.org/minlp/minlplib2/html/

Future Work:

- add more NLPs (from PrincetonLib, COCONUT, NEOS, ...)
- semi-automatic identification of duplicates
- more structure recognition, e.g., second-order cones
- define interesting subsets, especially a benchmark set for global solvers

Call for contributions:

- Contribute your own (MI)NLP instances! (Or send your model to minlp.org!)
- Ideally from a model for a real life problem.
- Also infeasible instances are welcomed.
- Any (well-known) format is good (e.g., AMPL, GAMS, ZIMPL, BARON, CPLEX LP, MPS, PIP, OSiL).
- MINLPLib instances are anonymized (scalar format using generic names).
- Your benefit: Solver developers may test and tune their solver for your problem.