(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.,

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

- 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   http://www.gamsworld.org

- 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
  INFORMS Journal on Computing 15, 114–119 (2003)
- "white-box" NLPs (GLOBALLib) and MINLPs (MINLPLib)

Firefox | MINLPLib Model Statistics

[MINLP World Home | Board | Solvers | MINLPLib | Links | GamsWorld Group | Search | Contact]

MINLPLib Model Statistics

Name	#Eqns	#Vars	#DVars	#NZ	#NNZ	BestInt	at Point
4stufen	99	150	48	319	87	116329.6706	p1
alan	8	9	4	24	3	2.9250	p1
batch	74	47	24	191	22	285506.5082	p1
batchdes	20	20	9	53	10	167427.6571	p1
beuster	115	158	52	398	159	116347.9503	p1
blendgap	360	332	66	1454	440	-19134.6103	p1
cecil_13	899	841	180	2812	360	-115656.4997	p2
chp_partload	2517	2249	45	6940	1916	23.5537	p1
contvar	285	297	88	1281	530	809149.8272	p1
csched1	23	77	63	174	8	-30639.2578	p1
csched1a	23	29	15	78	7	-30430.1770	p1
csched2	138	401	308	958	58	-166101.9964	p1
csched2a	138	233	140	622	57	-165398.7013	p2

- 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

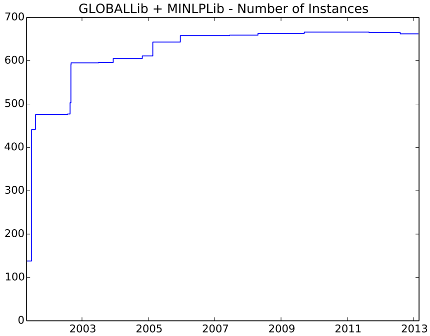
- scalar GAMS format

```
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

Show All entries

Search:

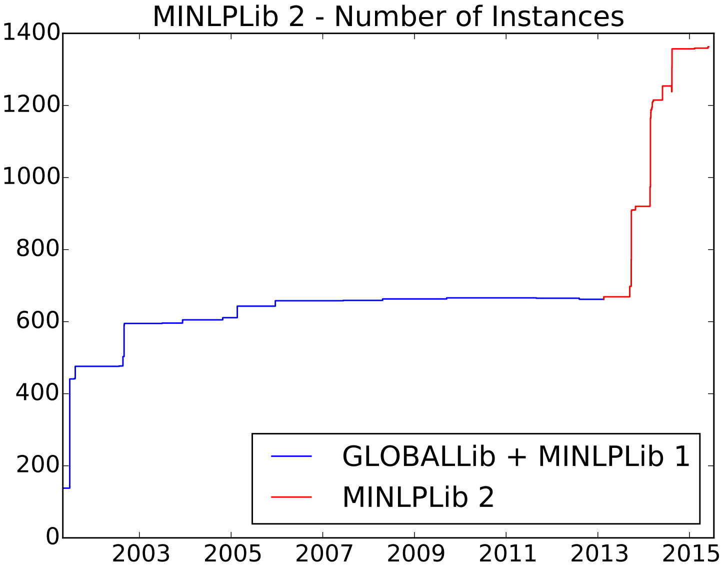
Name	Formats	Type	C	#Vars	#BinVars	#IntVars	#Cons	#SOS	#Semi	#NZ	CoefRange	S	Dual Bound	Primal Bound	Points
4stufen	gms mod nl osil	MBNLP	-	149	48		98			318	1.21e+11		102938.0658	116329.6706	p1
abel	gms lp mod nl osil pip	QP	*	30			14			100	2.86e+04	*	225.1946	225.1946	p1
alan	gms lp mod nl osil pip	MBQP	*	8	4		7			23	1.20e+01	*	2.9250	2.9250	p1
alkyl	gms mod nl osil	NLP	-	14			7			31	7.35e+03		-1.7650	-1.7650	p1
alkylation	gms mod nl osil	NLP	-	10			11			37	3.35e+05	*	1768.8073	1768.8070	p1
arki0001	gms lp mod nl osil pip	QP	*	1030			513			3813	4.32e+09	*	40.7129	40.7129	p1

Tasks:

- 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:
`http://www.gamsworld.org/minlp/minlplib2/html/index.html`.

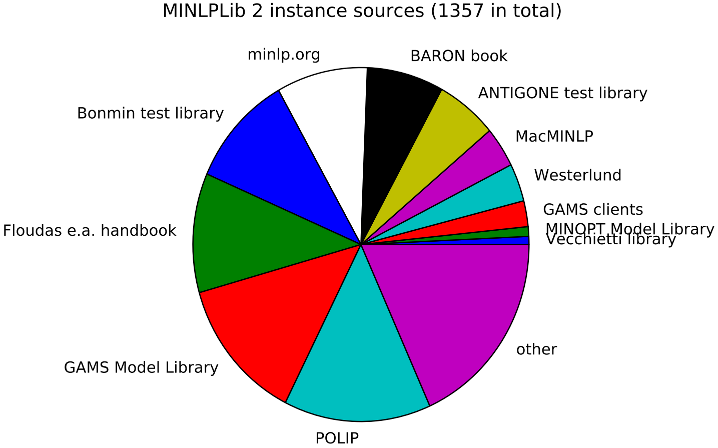
New NLP and MINLP Instances



Sources of newly added instances

Harvesting mainly from

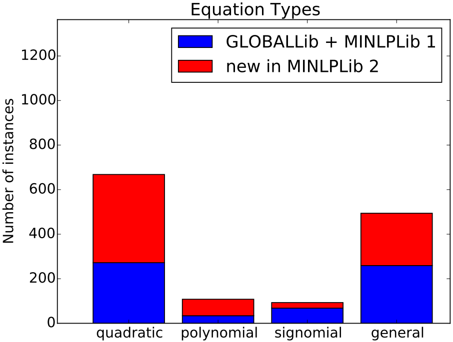
- CMU-IBM open source MINLP project (convex MINLPs)
- minlp.org
- POLIP (polynomial MINLPs)

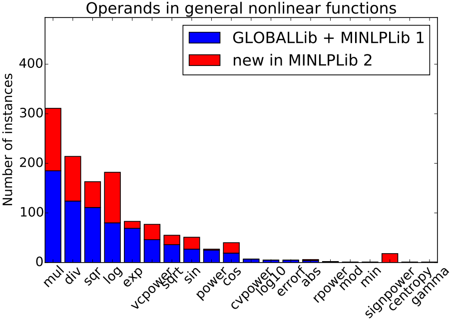


Instance Formats

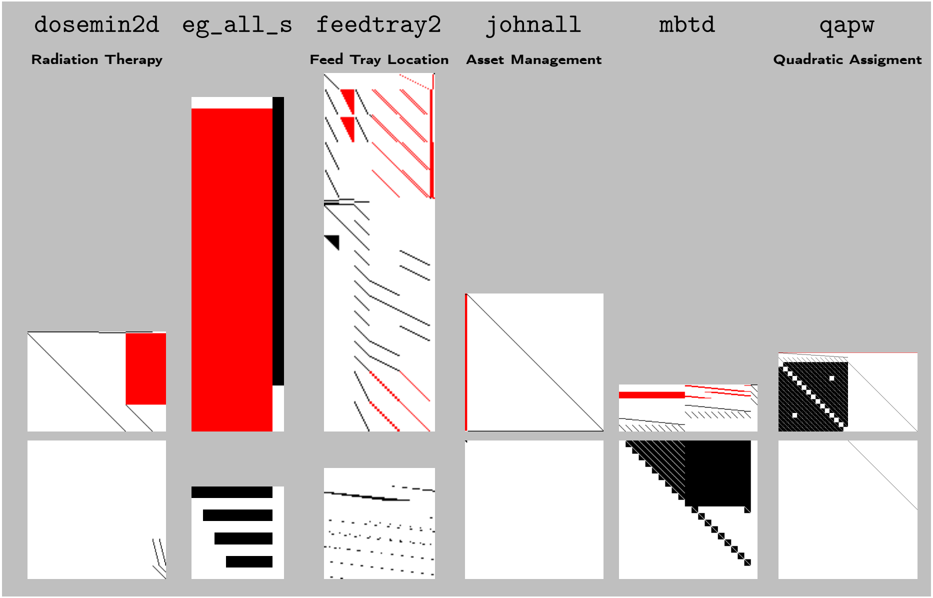
Format		#instances	
GAMS	`.gms`	**1363**	
AIMMS	`.ams`	1352	(no Gamma, latest additions missing)
AMPL	`.mod`	1337	(no errorf/signpower/Gamma/...)
AMPL	`.nl`	1331	(no errorf/signpower/Gamma/..., latest missing)
OSIL	XML	1342	(no signpower/Gamma/...)
CPLEX LP	`.lp`	667	(limited to quadratics)
PIP	`.pip`	770	(limited to polynomial)

Problem types





Sparsity Pattern – Examples

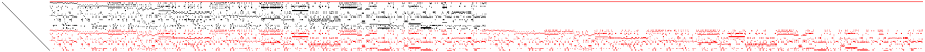


(top: Objective Gradient and Jacobian; bottom: Lagrangian Hessian)

Sparsity Pattern – Examples (cont.)

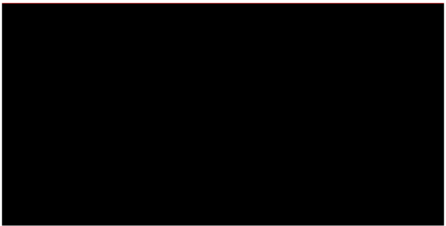
Jacobian `densitymod` **(Density modification based on single-crystal X-ray diffraction data; 23529 vars, 550 cons.)**

Jacobian `lop97ic` **(Rail Line Optimization, MIQCQP)**

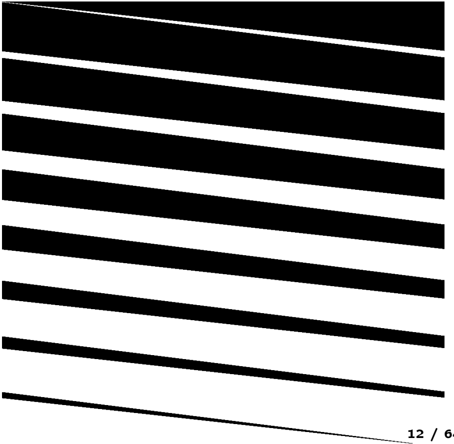


`milinfract` **(Solving Mixed-Integer Linear Fractional Programming Problems with Dinkelbach's Algorithm)**

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 $[\underline{x}, \bar{x}]$.
- 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:

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

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

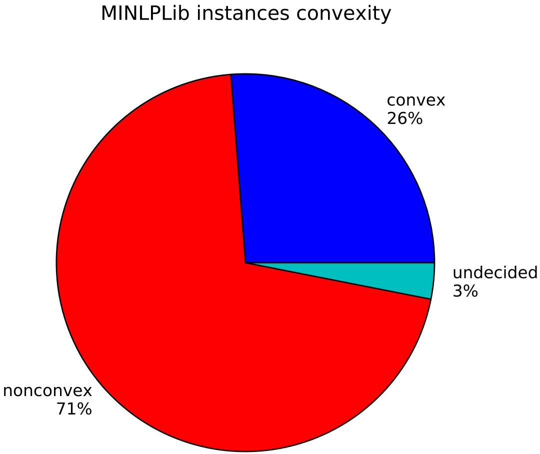
$$f(x) \text{ concave} \Rightarrow \log(f(x)) \text{ concave}$$

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

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.



Solution Points

MINLPLib instances traditionally come with known feasible solution points.

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

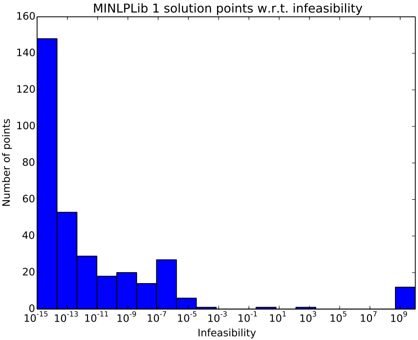
Feasibility checking:

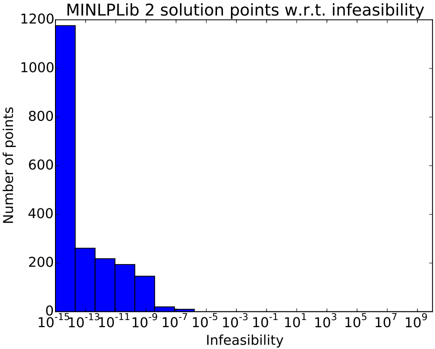
- compute maximal (unscaled) violation of constraints, variable bounds, and discreteness restrictions
- uses GAMS/EXAMINER2

Solution polishing: For a given point,

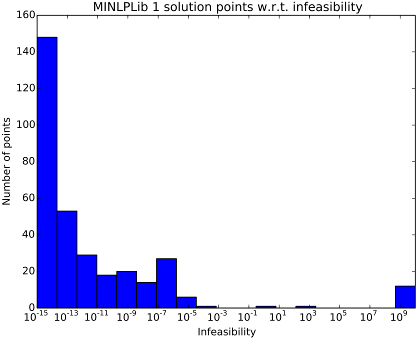
1. project onto variable bounds
2. round values for discrete variables to exact integers
3. ensure that semicontinuity/semiintegrality and special-ordered-set constraints are exactly satisfied
4. run CONOPT on MINLP with all binary/integer/semi*/SOS variables fixed, start from updated point, scaling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

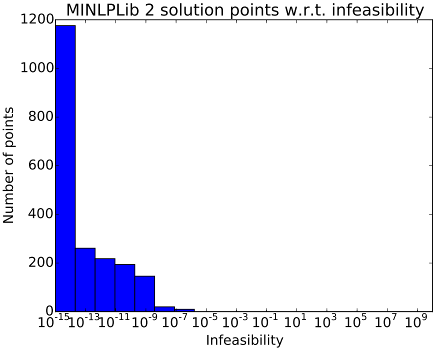
Polished Solution Points





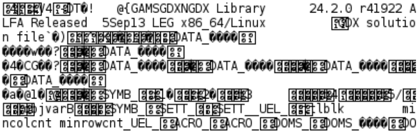
Polished Solution Points





Available in two formats:

GAMS Data Exchange (GDX)



ASCII (`.sol`)

```
x1     1.1180339887498901754
x2     1.3103706971044497739
b4     1.0000000000000000000
b5     1.0000000000000000000
objvar 7.6671800688131304113 4
```

Dual Bounds

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

Dual Bounds

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

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

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

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

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

Collected dual bounds from

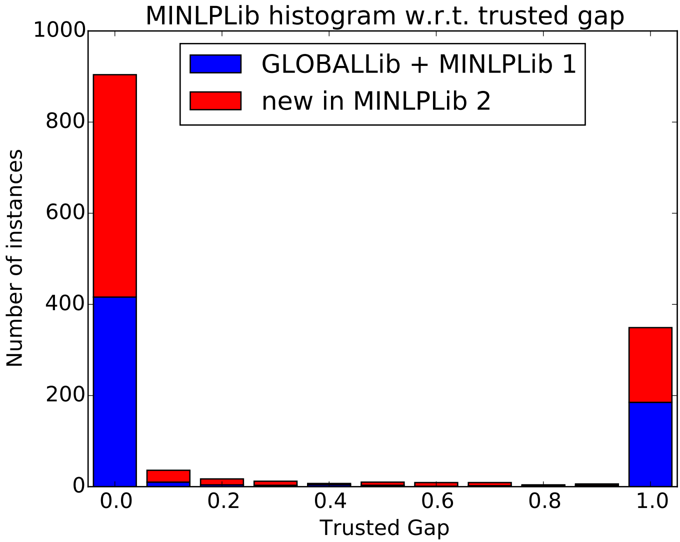
- solvers for general (MI)NLP (ANTIGONE, BARON, Couenne, Lindo, SCIP)
- solvers for convex MINLP on proven convex MINLPs (AlphaECP, Bonmin BB, Bonmin Hyb)

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



$0.0 \triangleq \leq 10^{-9}$ $\quad$ $1.0 \triangleq \geq 1.0$

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
                          nvars   ncons
jbearing100                5304       0
squfl030-150               4530    4650
watercontamination0202   106711  107209
watercontamination0303   107222  108217
```

- 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?

General Purpose Global Solvers Benchmark?

date	GAMS	ANTIGONE	BARON	COUENNE	LINDO	SCIP
07/15	24.5α	1.1	15.6.5	0.5	9.0.1983.157	3.2.0

• • •

Today: Go Columnwise

date	GAMS	ANTIGONE	BARON	COUENNE	LINDO	SCIP
08/11	23.7.3	–	**9.3**.1	**0.3**	**6.1**.1.588	–
04/12	23.8.2	–	**10.2**.0	**0.4**	**7.0**.1.421	**2.1**.1
11/12	23.9.5	–	**11.5**.2	0.4	7.0.1.497	2.1.2
02/13	24.0.2	–	**11.9**.1	0.4	7.0.1.497	**3.0**
07/13	24.1.3	**1.1**	**12.3**.3	0.4	**8.0**.1283.385	3.0
05/14	24.2.3	1.1	**12.7**.7	0.4	8.0.1694.498	3.0
09/14	24.3.3	1.1	**14.0**.3	0.4	8.0.1694.550	**3.1**
06/15	24.4.6	1.1	**14.4**.0	0.4	**9.0**.1983.157	3.1
07/15	24.5α	1.1	**15.6**.5	**0.5**	9.0.1983.157	**3.2**.0

- 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
$\Rightarrow$ 40 solver versions $\times$ 1363 instances = 54520 runs (!)

Which instances to run?

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

- ✓ large number of instances
  $\Rightarrow$ 40 solver versions $\times$ 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
  $\Rightarrow$ 40 solver versions $\times$ 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
  - ...

Which instances to run?

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

- ✓ large number of instances
  $\Rightarrow$ 40 solver versions $\times$ 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
  $\Rightarrow$ 40 solver versions $\times$ 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
  $\Rightarrow$ 40 solver versions $\times$ 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.

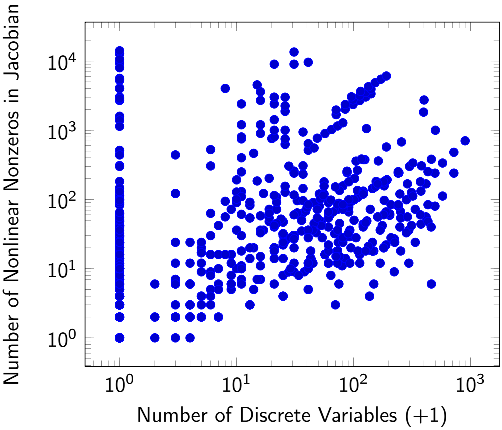
> **MIPLIB 2010**
>
> **Mixed Integer Programming Library version 5**
>
> Thorsten Koch · Tobias Achterberg · Erling Andersen · Oliver Bastert · Timo Berthold · Robert E. Bixby · Emilie Danna · Gerald Gamrath · Ambros M. Gleixner · Stefan Heinz · Andrea Lodi · Hans Mittelmann · Ted Ralphs · Domenico Salvagnin · Daniel E. Steffy · Kati Wolter

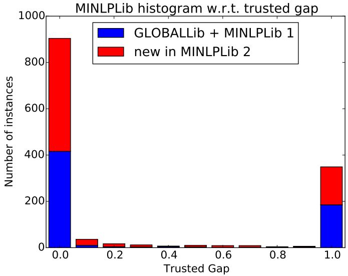
But with 3 weeks until ISMP: Apply the P.I.T.T. heuristic.

Prune Instances by Tractability and Triviality Heuristic

1. Remove intractable instances

- consider only the 881 instances that are marked as solved in MINLPLib 2

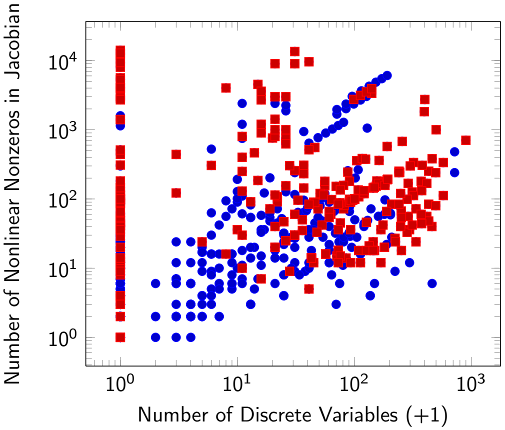




Prune Instances by Tractability and Triviality Heuristic

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:

alkylation	clay0304h	ex6_1_1	fo9_ar5_1	house
arki0003	clay0305h	ex6_1_3	gasnet	jbearing25
arki0005	crudeoil_lee2_10	ex6_2_12	genpooling_meyer04	jbearing75
arki0006	crudeoil_lee3_07	ex6_2_14	ghg_1veh	johnall
arki0019	crudeoil_lee3_08	ex6_2_8	ghg_3veh	kall_circles_c6a
arki0024	crudeoil_lee3_09	ex6_2_9	glider100	kall_circles_c6b
autocorr_bern20-10	crudeoil_lee3_10	ex7_2_4	graphpart_2g-0066-0066	kall_circles_c7a
autocorr_bern20-15	crudeoil_li06	ex8_1_7	graphpart_2g-0077-0077	kall_circles_c8a
autocorr_bern25-06	csched1a	ex8_2_1b	graphpart_2g-0088-0088	kall_circlespolygons_c1p
autocorr_bern25-13	edgecross10-060	ex8_2_4b	graphpart_2g-0099-9211	kall_circlespolygons_c1p
autocorr_bern30-04	edgecross10-070	ex8_4_1	graphpart_2pm-0066-0066	kall_circlesrectangles_c
autocorr_bern35-04	edgecross10-080	ex8_4_3	graphpart_2pm-0077-0777	kall_circlesrectangles_c
batch0812_nc	edgecross14-039	ex8_4_4	graphpart_2pm-0088-0888	kall_congruentcircles_c7
batchs201210m	edgecross14-058	ex8_4_5	graphpart_2pm-0099-0999	kall_diffcircles_10
bayes2_50	edgecross14-078	ex8_4_8_bnd	graphpart_3g-0334-0334	kall_diffcircles_5b
blend480	edgecross14-176	filter	graphpart_3g-0344-0344	kall_diffcircles_7
blend531	edgecross20-040	fin2bb	graphpart_3g-0444-0444	kall_diffcircles_8
blend718	edgecross22-048	flay05h	graphpart_3pm-0244-0244	kall_diffcircles_9
blend852	emfl050_5_5	fo7	graphpart_3pm-0333-0333	launch
carton7	emfl100_5_5	fo8	graphpart_3pm-0334-0334	lop97icx
casctanks	ethanolh	fo8_ar25_1	graphpart_3pm-0344-0344	mathopt5_7
cecil_13	ethanolm	fo8_ar2_1	graphpart_3pm-0444-0444	mathopt5_8
chem	ex1252a	fo9	graphpart_clique-30	mhw4d
clay0203h	ex14_1_1	fo9_ar25_1	graphpart_clique-40	milinfract
clay0204h	ex14_1_7	fo9_ar2_1	gsg_0001	minlphix
clay0205h	ex4_1_5	fo9_ar3_1	hda	minsurf100
clay0303h	ex4_1_6	fo9_ar4_1	heatexch_trigen	...

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_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
- crudeoil_lee2_10
- crudeoil_lee3_07
- crudeoil_lee3_08
- crudeoil_lee3_09
- crudeoil_lee3_10
- crudeoil_li06
- csched1a
- edgecross10-060
- edgecross10-070
- edgecross10-080
- edgecross14-039
- edgecross14-058
- edgecross14-078
- edgecross14-176
- edgecross20-040
- edgecross22-048
- emfl050_5_5
- emfl100_5_5
- ethanolh
- ethanolm
- ex1252a
- ex14_1_1
- ex14_1_7
- ex4_1_5
- ex4_1_6
- 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
- filter
- fin2bb
- flay05h
- fo7
- fo8
- fo8_ar25_1
- fo8_ar2_1
- fo9
- fo9_ar25_1
- fo9_ar2_1
- fo9_ar3_1
- fo9_ar4_1
- fo9_ar5_1
- gasnet
- genpooling_meyer04
- ghg_1veh
- ghg_3veh
- glider100
- graphpart_2g-0066-0066
- graphpart_2g-0077-0077
- graphpart_2g-0088-0088
- graphpart_2g-0099-9211
- graphpart_2pm-0066-0066
- graphpart_2pm-0077-0777
- graphpart_2pm-0088-0888
- graphpart_2pm-0099-0999
- graphpart_3g-0334-0334
- graphpart_3g-0344-0344
- graphpart_3g-0444-0444
- graphpart_3pm-0244-0244
- graphpart_3pm-0333-0333
- graphpart_3pm-0334-0334
- graphpart_3pm-0344-0344
- graphpart_3pm-0444-0444
- graphpart_clique-30
- graphpart_clique-40
- gsg_0001
- hda
- heatexch_trigen
- house
- jbearing25
- jbearing75
- johnall
- kall_circles_c6a
- kall_circles_c6b
- kall_circles_c7a
- kall_circles_c8a
- kall_circlespolygons_c1p
- kall_circlespolygons_c1p
- kall_circlesrectangles_c
- kall_circlesrectangles_c
- kall_congruentcircles_c7
- kall_diffcircles_10
- kall_diffcircles_5b
- kall_diffcircles_7
- kall_diffcircles_8
- kall_diffcircles_9
- launch
- lop97icx
- mathopt5_7
- mathopt5_8
- mhw4d
- milinfract
- minlphix
- minsurf100
- ...

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~~
crudeoil_lee2_10
~~crudeoil_lee3_07~~
~~crudeoil_lee3_08~~
~~crudeoil_lee3_09~~
~~crudeoil_lee3_10~~
~~crudeoil_li06~~
csched1a
edgecross10-060
~~edgecross10-070~~
~~edgecross10-080~~
~~edgecross14-039~~
~~edgecross14-058~~
~~edgecross14-078~~
~~edgecross14-176~~
~~edgecross20-040~~
~~edgecross22-048~~
emfl050_5_5
~~emfl100_5_5~~
ethanolh
~~ethanolm~~
ex1252a
ex14_1_1
~~ex14_1_7~~
ex4_1_5
~~ex4_1_6~~
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~~
filter
fin2bb
flay05h
fo7
fo8
fo8_ar25_1
~~fo8_ar2_1~~
fo9
fo9_ar25_1
~~fo9_ar2_1~~
~~fo9_ar3_1~~
~~fo9_ar4_1~~
~~fo9_ar5_1~~
gasnet
genpooling_meyer04
ghg_1veh
ghg_3veh
glider100
graphpart_2g-0066-0066
~~graphpart_2g-0077-0077~~
~~graphpart_2g-0088-0088~~
~~graphpart_2g-0099-9211~~
~~graphpart_2pm-0066-0066~~
~~graphpart_2pm-0077-0777~~
~~graphpart_2pm-0088-0888~~
~~graphpart_2pm-0099-0999~~
~~graphpart_3g-0334-0334~~
~~graphpart_3g-0344-0344~~
~~graphpart_3g-0444-0444~~
~~graphpart_3pm-0244-0244~~
~~graphpart_3pm-0333-0333~~
~~graphpart_3pm-0334-0334~~
~~graphpart_3pm-0344-0344~~
~~graphpart_3pm-0444-0444~~
~~graphpart_clique-30~~
~~graphpart_clique-40~~
gsg_0001
hda
heatexch_trigen
house
jbearing25
~~jbearing75~~
johnall
kall_circles_c6a
~~kall_circles_c6b~~
~~kall_circles_c7a~~
~~kall_circles_c8a~~
~~kall_circlespolygons_c1p~~
~~kall_circlespolygons_c1p~~
~~kall_circlesrectangles_c~~
~~kall_circlesrectangles_c~~
~~kall_congruentcircles_c7~~
kall_diffcircles_10
~~kall_diffcircles_5b~~
~~kall_diffcircles_7~~
~~kall_diffcircles_8~~
~~kall_diffcircles_9~~
launch
lop97icx
mathopt5_7
~~mathopt5_8~~
mhw4d
milinfract
minlphix
minsurf100
...

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

In summary:

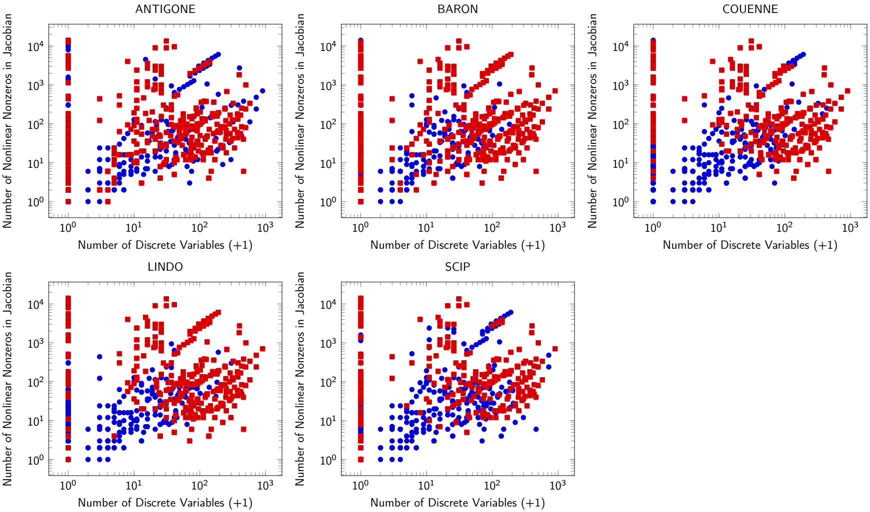
1. Keep only instances that are marked as solved in MINLPLib 2.
2. Keep only instances that take $\geq$ 60s with oldest version of solver and that can be handled by solver.
3. Reduce instances with similar names.

For SCIP, this reduces from 1363 to 881 to 123 instances:

alkylation
arki0003
autocorr_bern20-10
batch0812_nc
batchs201210m
bayes2_50
blend480
blend531
blend718
blend852
carton7
casctanks
cecil_13
chem
clay0203h
clay0303h
crudeoil_lee2_10
csched1a
edgecross10-060
emfl050_5_5
emfl100_5_5
ethanolh
ex1252a
ex14_1_1
ex4_1_5
ex6_1_1
ex6_2_12
ex7_2_4
ex8_1_7
ex8_2_1b
ex8_4_1
filter
fin2bb
flay05h
fo7
fo8
fo8_ar25_1
fo9
fo9_ar25_1
gasnet
genpooling_meyer04
ghg_1veh
ghg_3veh
glider100
graphpart_2g-0066-0066
gsg_0001
hda
heatexch_trigen
house
jbearing25
johnall
kall_circles_c6a
kall_diffcircles_10
launch
lop97icx
mathopt5_7
mhw4d
milinfract
minlphix
minsurf100
multiplants_mtg1a
no7_ar3_1
nous1
nvs09
nvs22
o7
o7_2
o7_ar25_1
o7_ar3_1
o7_ar4_1
o7_ar5_1
o8_ar4_1
o9_ar4_1
oil
oil2
parallel
pinene50
pointpack08
pooling_epa1
prob07
process
procsyn
prolog
qp3
routingdelay_bigm
rsyn0805m02h
sepasequ_convent
sfacloc2_2_80
slay07h
slay09h
slay10h
smallinvDAXr1b020-022
sporttournament14
squfl010-025
sssd08-04
sssd12-05
sssd15-04
sssd16-07
sssd18-06
sssd20-04
sssd25-04
st_e35
stockcycle
supplychai
syn10m03h
syn15m02h
syn20m02h
syn30h
syn30m02h
syn40h
syn40m02h

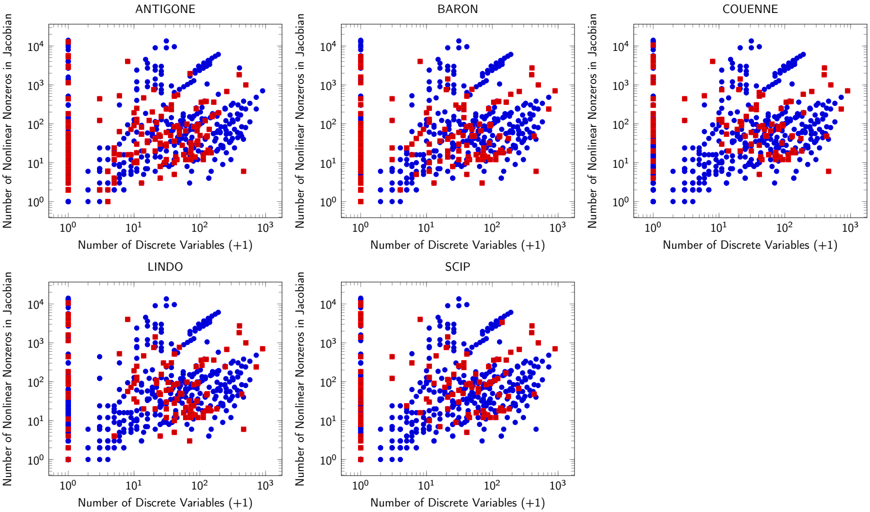
PITT test set for each solver

Removed easy and unsolvable instances:



PITTED test set for each solver

Removed easy and unsolvable instances, then filter by name:



Run jobs

date	GAMS	ANTIGONE	BARON	COUENNE	LINDO	SCIP
08/11	23.7.3	–	**9.3**.1	**0.3**	**6.1**.1.588	–
04/12	23.8.2	–	**10.2**.0	**0.4**	**7.0**.1.421	**2.1**.1
11/12	23.9.5	–	**11.5**.2	0.4	7.0.1.497	2.1.2
02/13	24.0.2	–	**11.9**.1	0.4	7.0.1.497	**3.0**
07/13	24.1.3	**1.1**	**12.3**.3	0.4	**8.0**.1283.385	3.0
05/14	24.2.3	1.1	**12.7**.7	0.4	8.0.1694.498	3.0
09/14	24.3.3	1.1	**14.0**.3	0.4	8.0.1694.550	**3.1**
06/15	24.4.6	1.1	**14.4**.0	0.4	**9.0**.1983.157	3.1
07/15	24.5α	1.1	**15.6**.5	**0.5**	9.0.1983.157	**3.2**.0

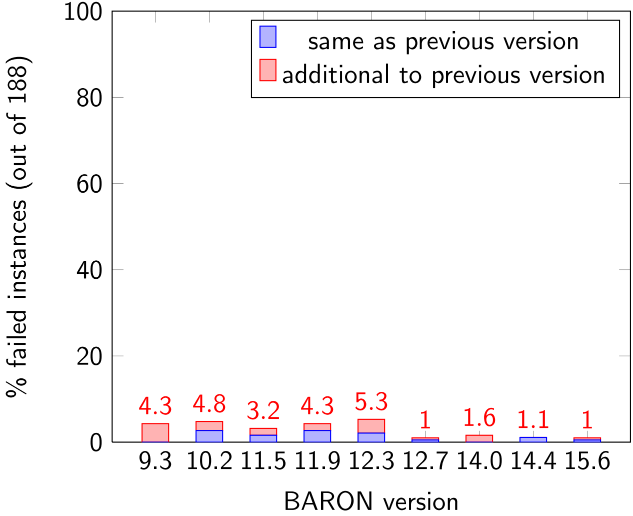
```
for GAMS in $GAMSS ; do
 for SOLVER in $SOLVERS($GAMS) ; do
  for INSTANCE in $TESTSET($SOLVER) ; do
   sbatch --exclusive --time=0:1800 $GAMS $INSTANCE SOLVER=$SOLVER
  done
 done
done
```

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

BARON: Fails

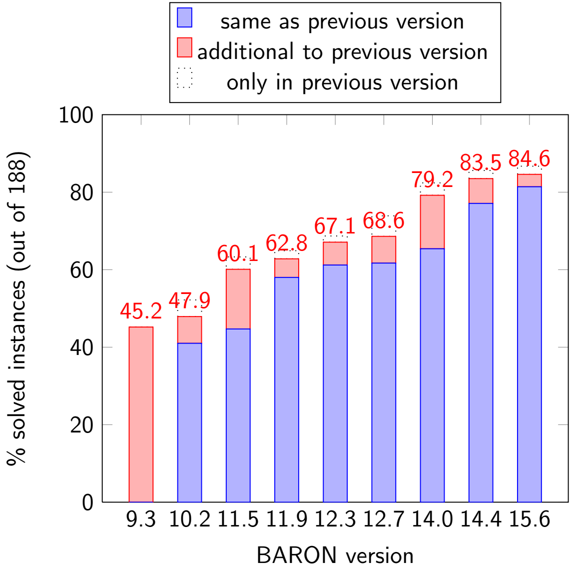
A solver failed, if it

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

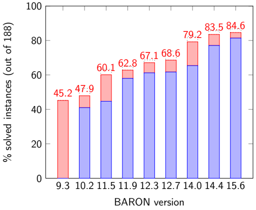


BARON: Solved

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



BARON: Solved – What happened?



From the release notes:

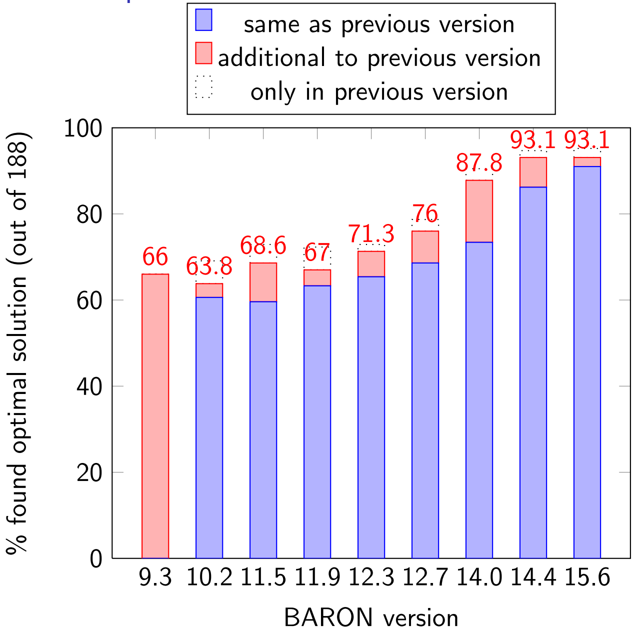
11.0: "This version comes with a wealth of new branching, relaxation, convexity exploitation, local search, and range reduction techniques."

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

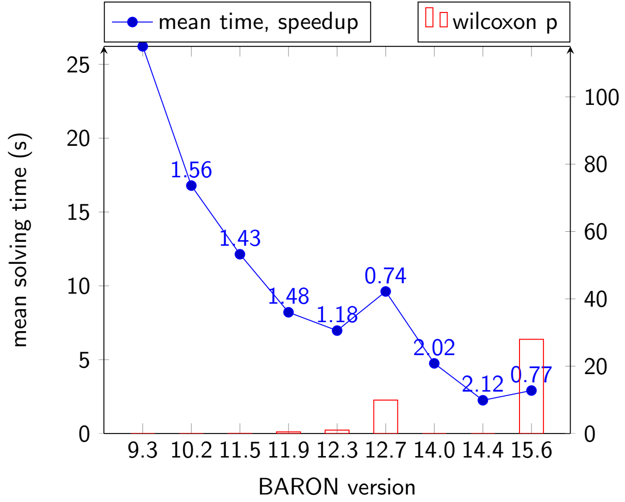
12.3: "New relaxations for certain types of quadratic problems", "Improved integer presolve", "Incorporation of convex envelopes for certain low-dimensional functions"

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



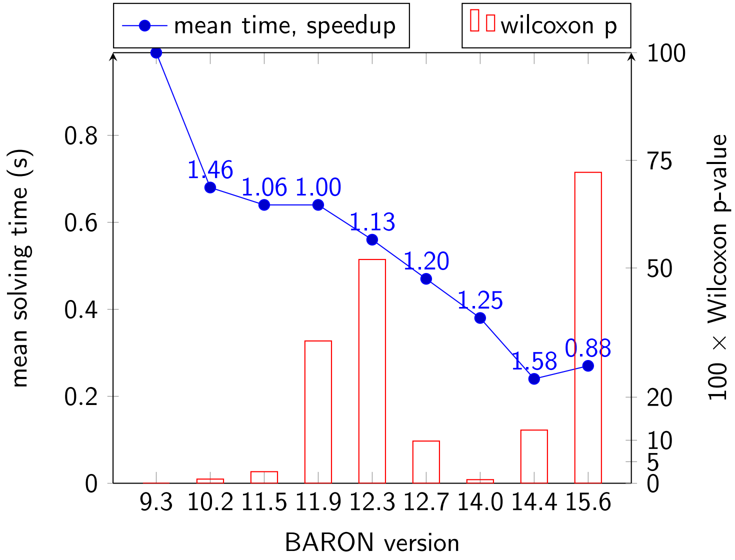
BARON: Solving time on instances that never failed (163)



Overall speedup: 9.00

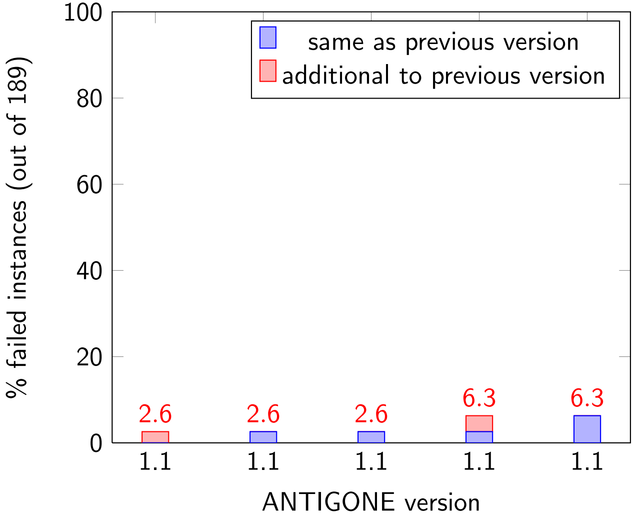
12.7: "Automatic setting of many options based on problem characteristics and learning algorithms."

BARON: Solving time on instances solved by all vers. (69)

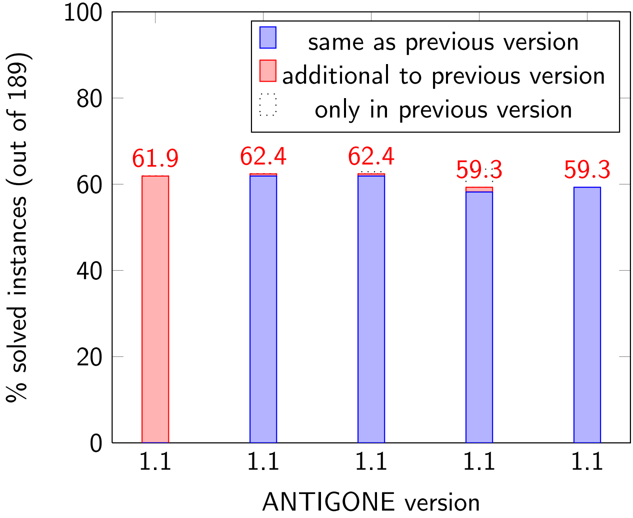


Overall speedup: 3.67

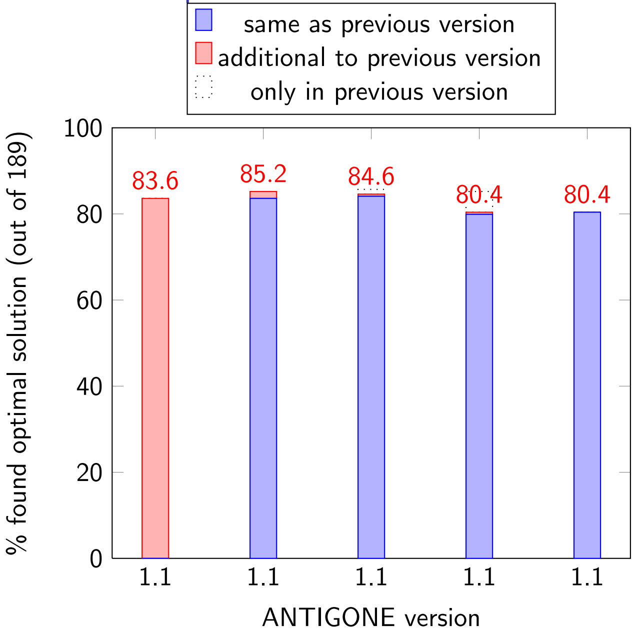
ANTIGONE: Fails



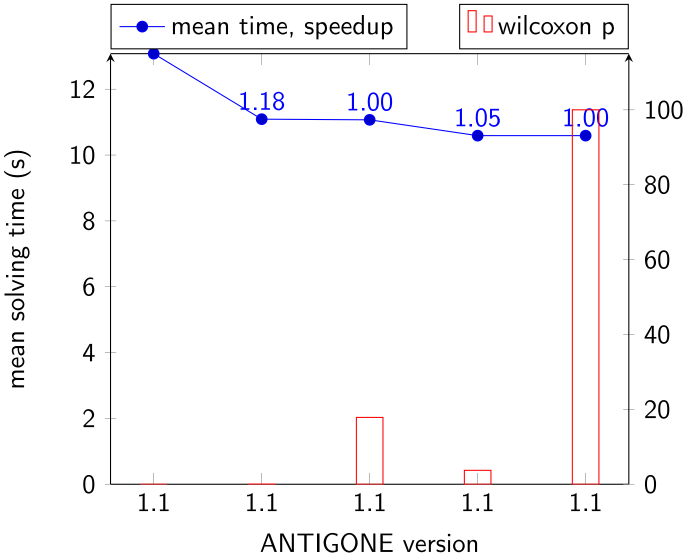
ANTIGONE: Solved



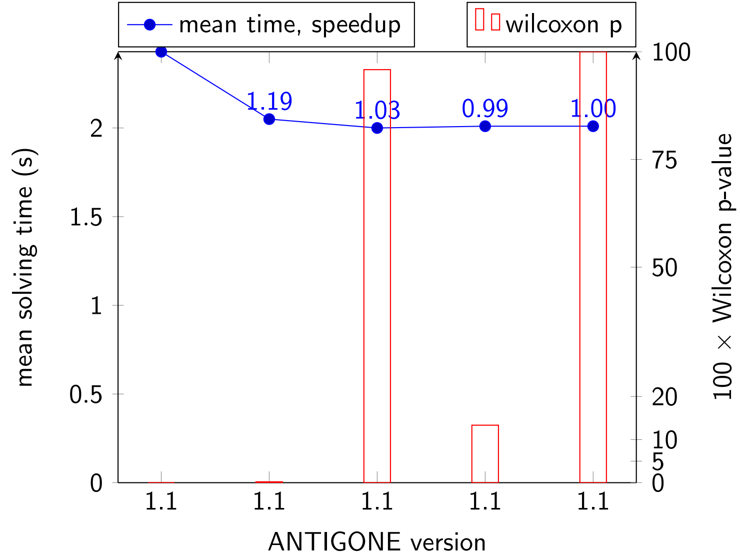
ANTIGONE: Found optimal solution



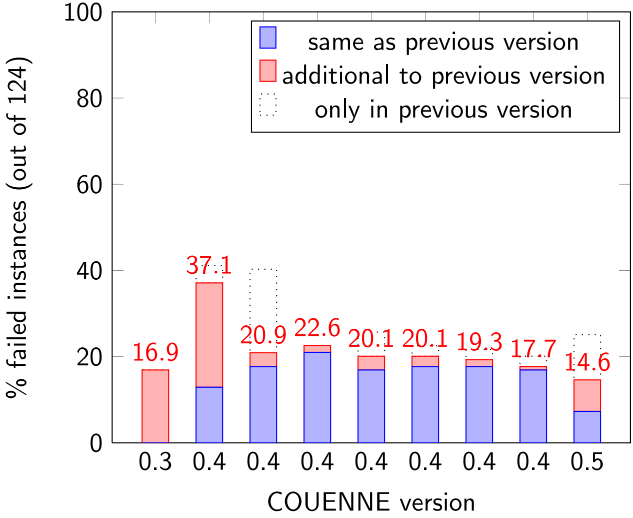
ANTIGONE: Solving time on instances that never failed (177)



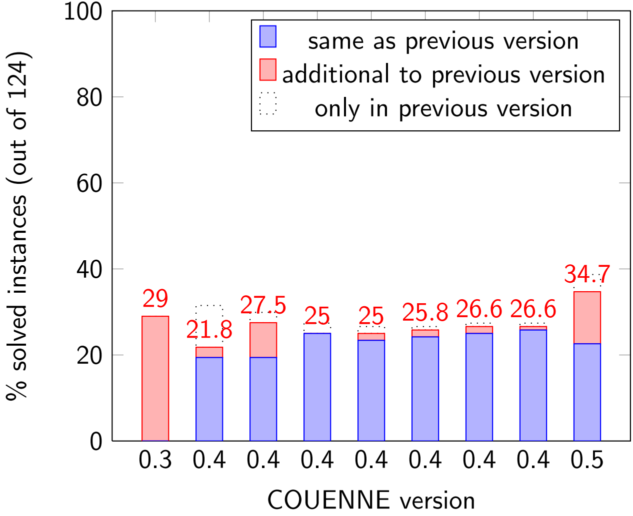
ANTIGONE: Solving time on instances solved by all (110)



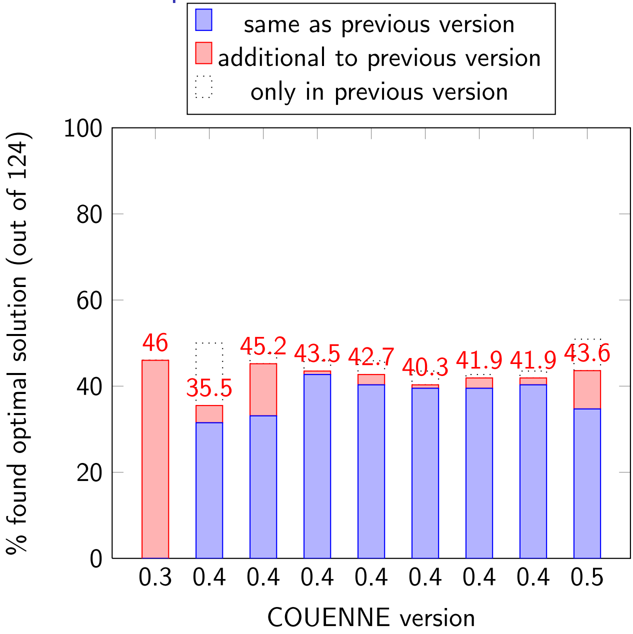
COUENNE: Fails



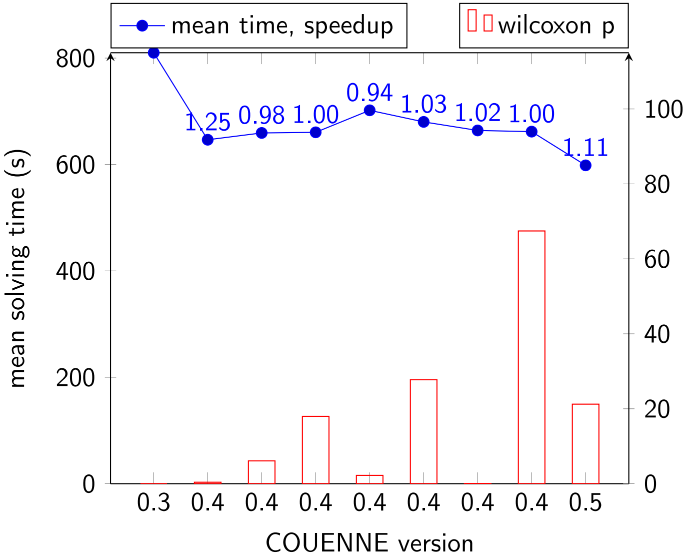
COUENNE: Solved



COUENNE: Found optimal solution



COUENNE: Solving time on instances that never failed (65)



Overall speedup: 1.35

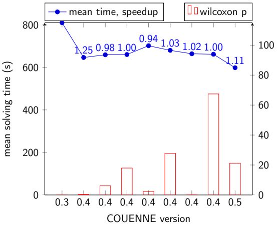
COUENNE: Solving time on instances that never failed (65)

[Couenne] Couenne stable release 0.4

Pietro Belotti [pbelott at clemson.edu]
Mon Aug 8 05:13:15 EDT 2011

- Next message: [Couenne] stable/0.4 does not compile
- **Messages sorted by:** [date] [thread] [subject] [author]

```
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 >= f(x) or y
<= 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.

5) "Two-Implied bound tightening", a new bound reduction procedure
described in http://www.optimization-online.org/DB_FILE/2011/02/2931.pdf

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
2. Orbital Branching
3. Fixed Point BT
4. “semi-auxiliaries”
5. Two-Implied BT
6. various bug fixes

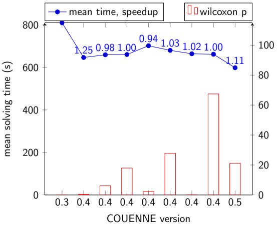
COUENNE: Solving time on instances that never failed (65)

[Couenne] Couenne stable release 0.4

Pietro Belotti pbelott at clemson.edu
Mon Aug 8 05:13:15 EDT 2011

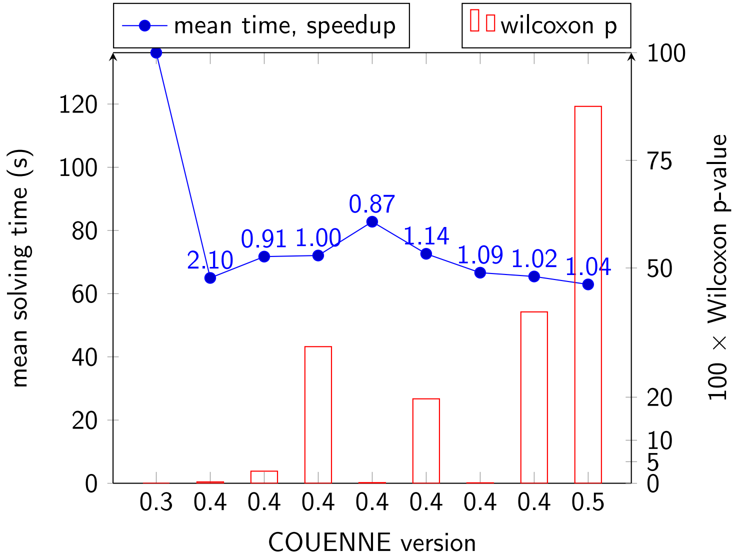
- Next message: [Couenne] stable/0.4 does not compile
- **Messages sorted by:** [date] [thread] [subject] [author]

```
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 >= f(x) or y
<= 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.

5) "Two-Implied bound tightening", a new bound reduction procedure
described in http://www.optimization-online.org/DB_FILE/2011/02/2931.pdf

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
```



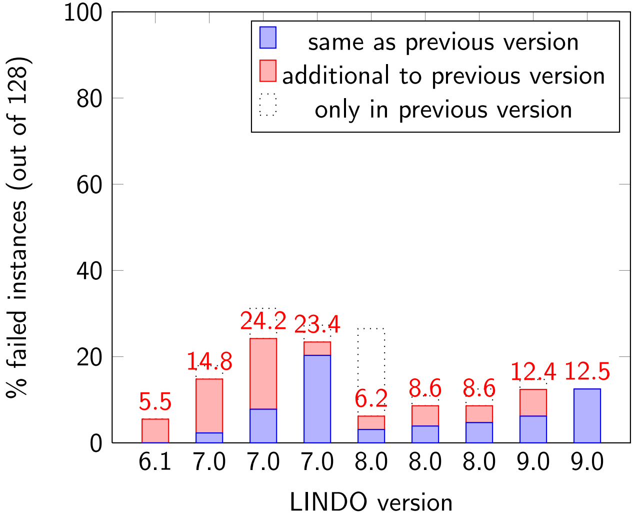
1. Feasibility Pump
   `feasibility_pump no`
2. Orbital Branching
   `orbital_branching no`
3. Fixed Point BT
   `fixpoint_bt 0`
4. "semi-auxiliaries"
   `use_semiaux yes`
5. Two-Implied BT
   `two_implied_bt 0`
6. various bug fixes

COUENNE: Solving time on instances solved by all (16)



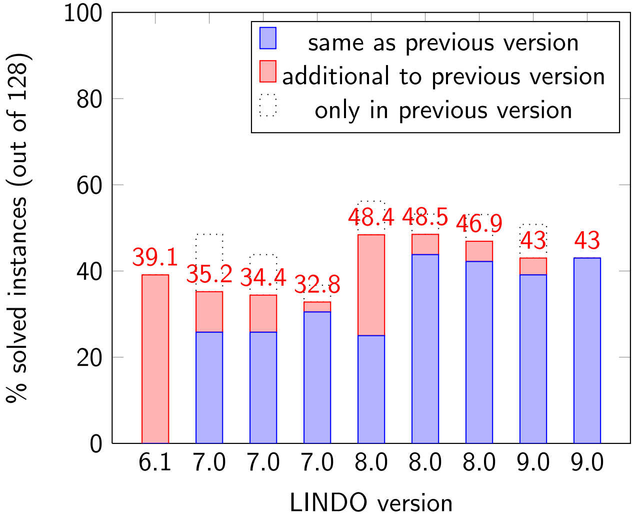
Overall speedup: 2.16

LINDO: Fails



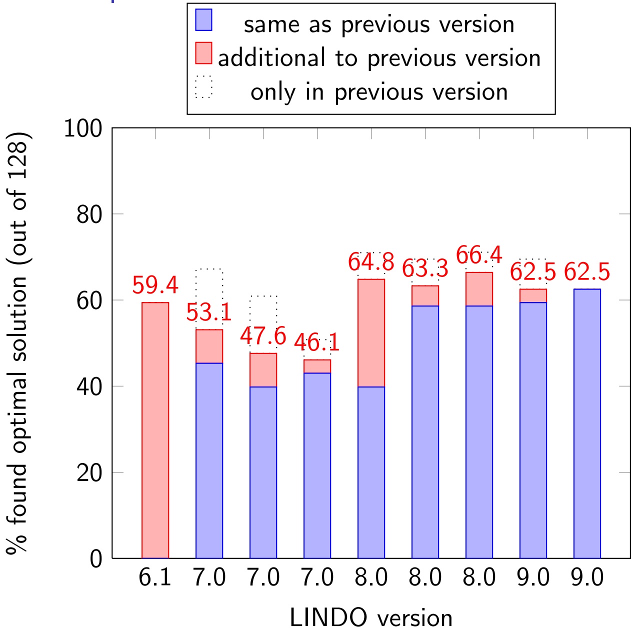
100
80
60
40
20
0
% failed instances (out of 128)
same as previous version
additional to previous version
only in previous version
5.5
14.8
24.2
23.4
6.2
8.6
8.6
12.4
12.5
6.1
7.0
7.0
7.0
8.0
8.0
8.0
9.0
9.0
LINDO version

LINDO: Solved

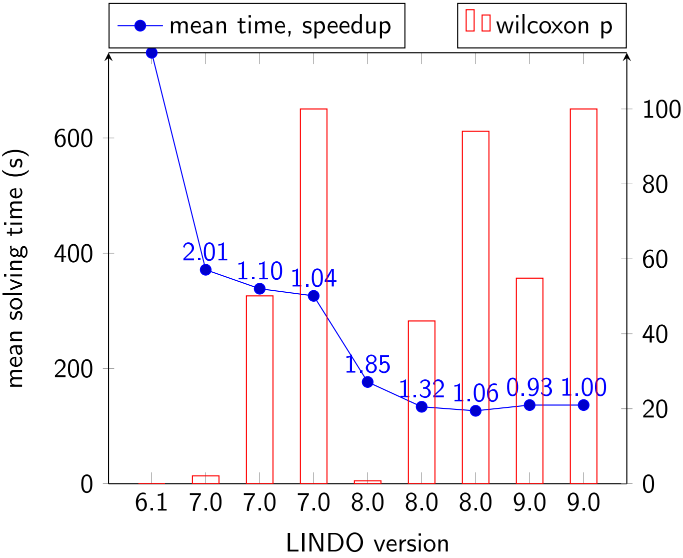


LINDO 8.0: improvements in primal heuristics for MIP (feas. pump) and nonconvex NLP (multistart)

LINDO: Found optimal solution

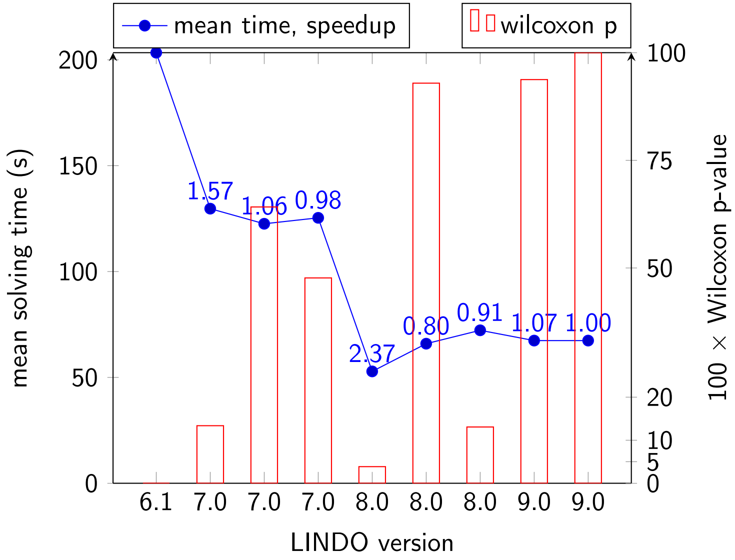


LINDO: Solving time on instances that never failed (72)



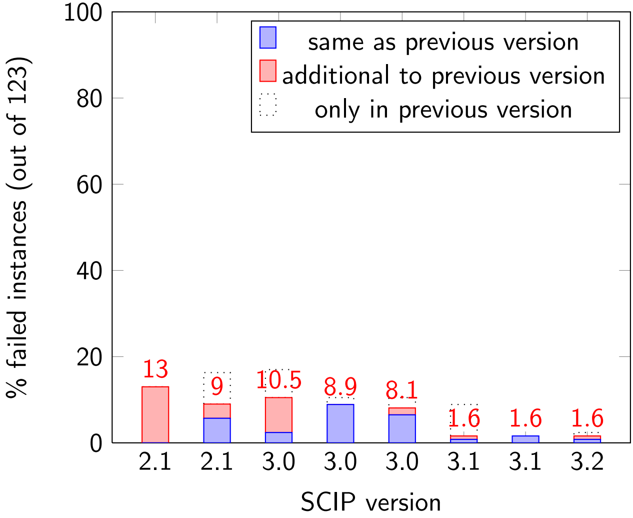
Overall speedup: 5.48

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



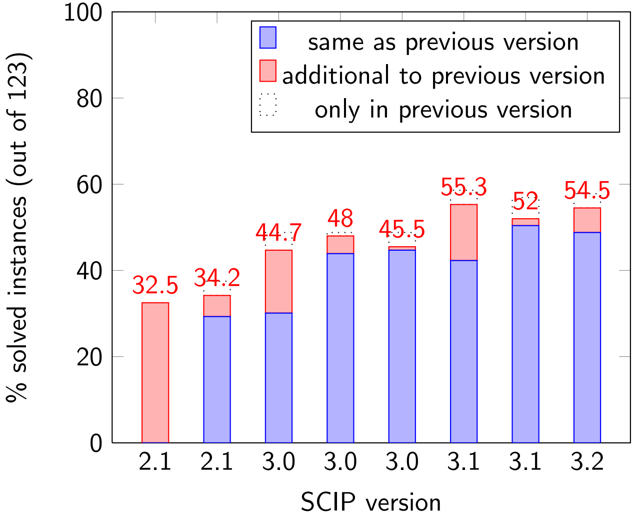
Overall speedup: 3.02

SCIP: Fails



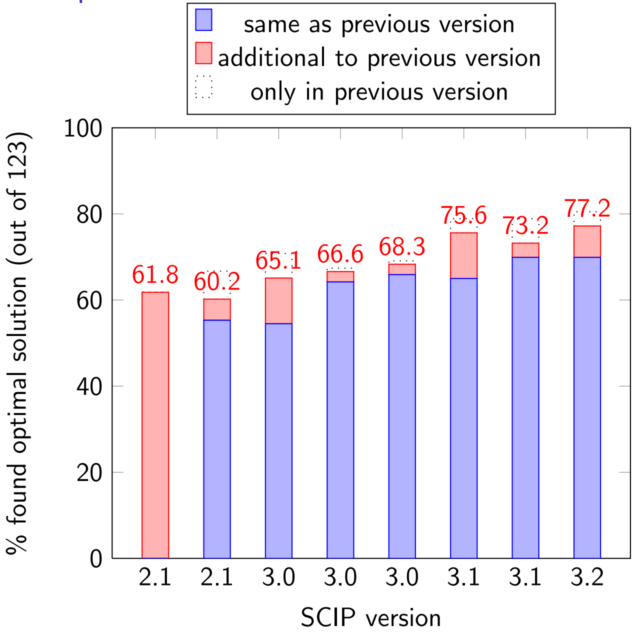
100
80
60
40
20
0
% failed instances (out of 123)
same as previous version
additional to previous version
only in previous version
13
9
10.5
8.9
8.1
1.6
1.6
1.6
2.1
2.1
3.0
3.0
3.0
3.1
3.1
3.2
SCIP version

SCIP: Solved

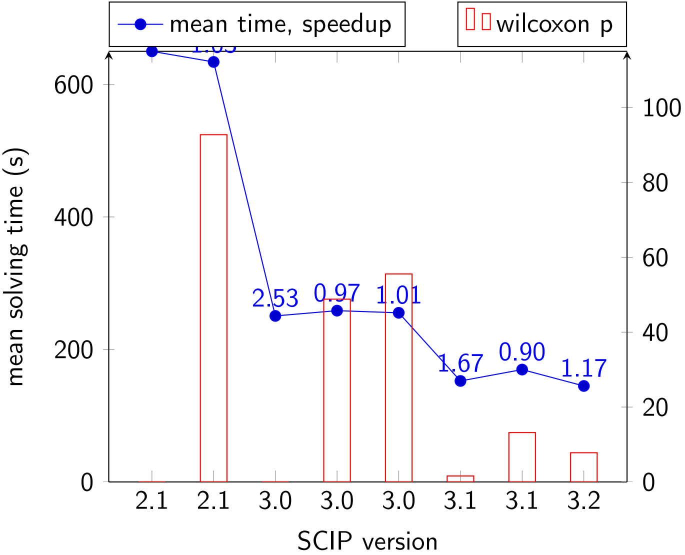


same as previous version
additional to previous version
only in previous version
% solved instances (out of 123)
100
80
60
40
20
0
32.5
34.2
44.7
48
45.5
55.3
52
54.5
2.1
2.1
3.0
3.0
3.0
3.1
3.1
3.2
SCIP version

SCIP: Found optimal solution

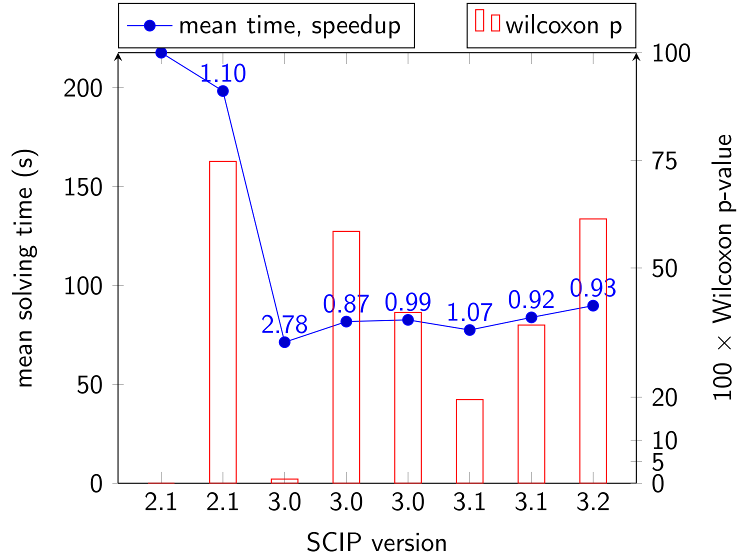


SCIP: Solving time on instances that never failed (96)



Overall speedup: 4.49

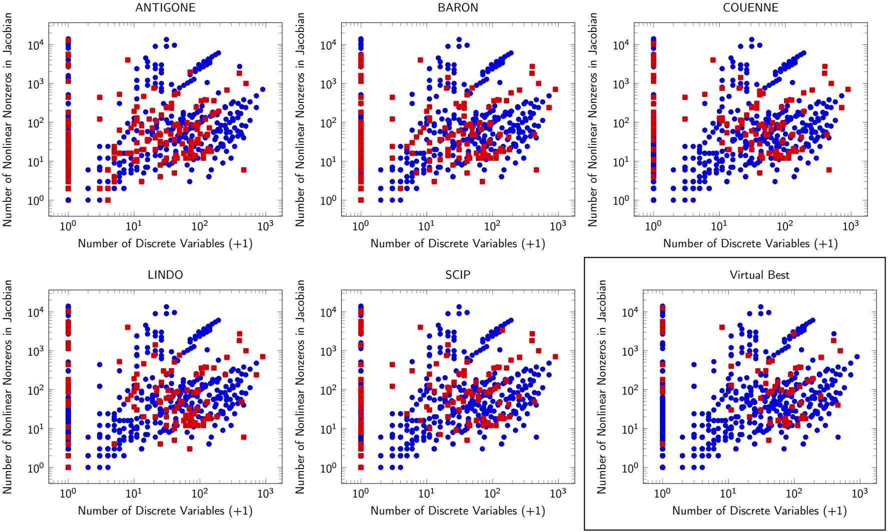
SCIP: Solving time on instances solved by all vers. (31)



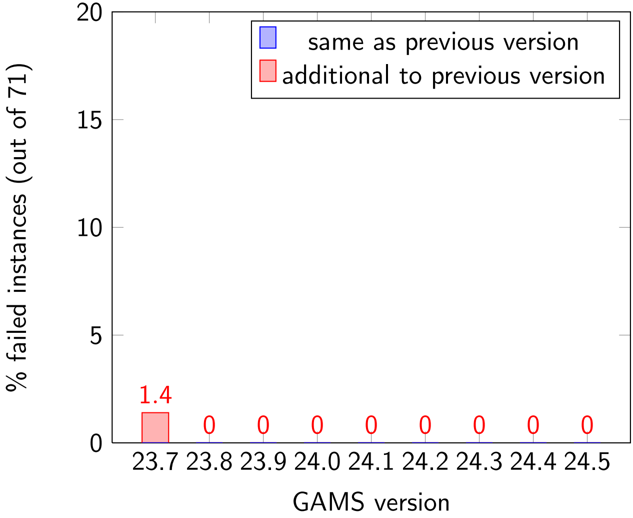
Overall speedup: 2.42

"Virtual Best" Solver

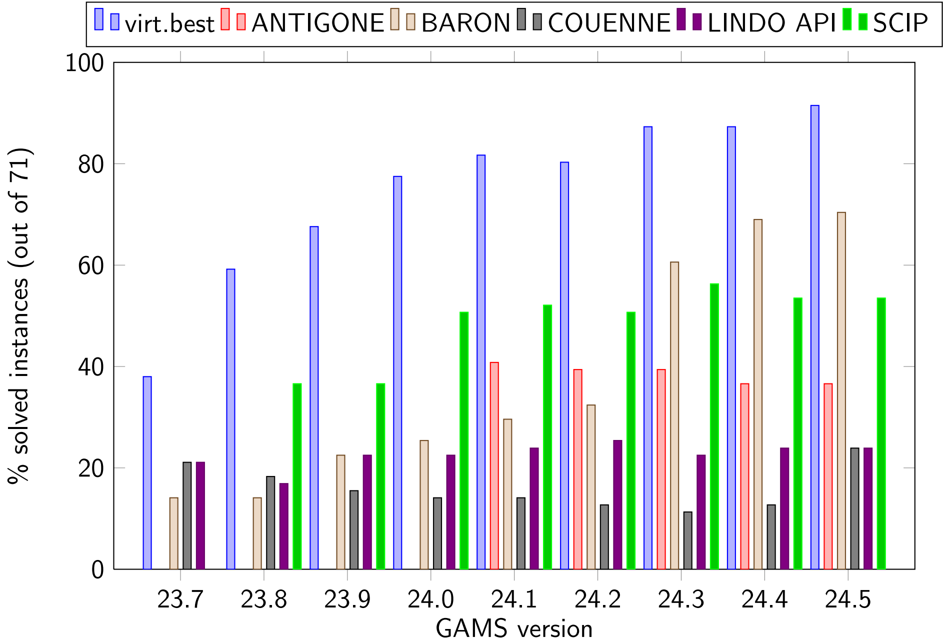
- common subset of instances
- for each instance and GAMS version, pick best results among all solvers



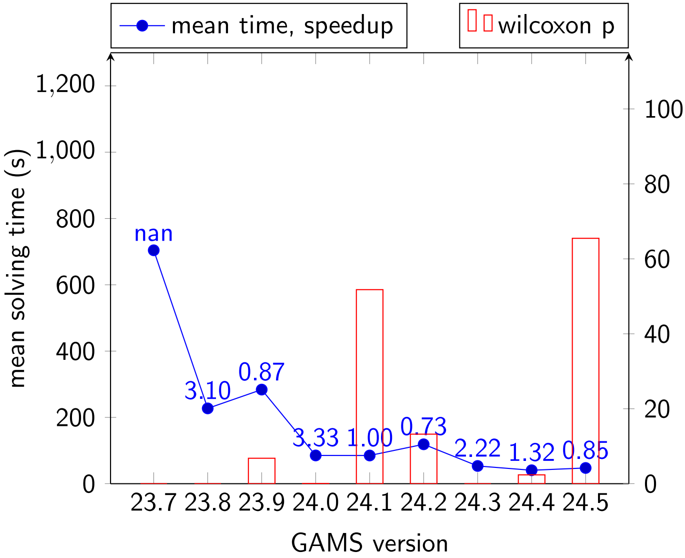
Virtual Best: Fails



Virtual Best: Solved

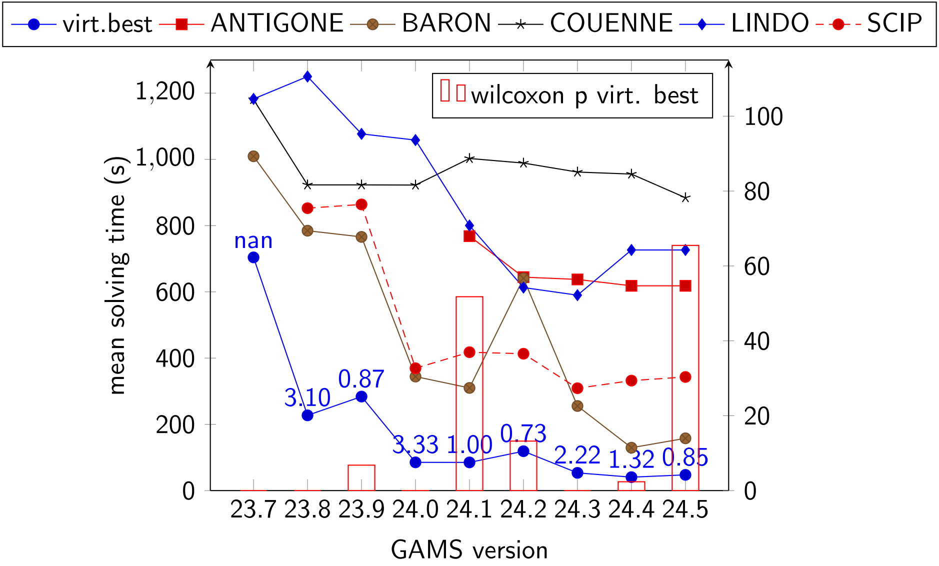


Virtual Best: Solving time on instances that never failed (70)



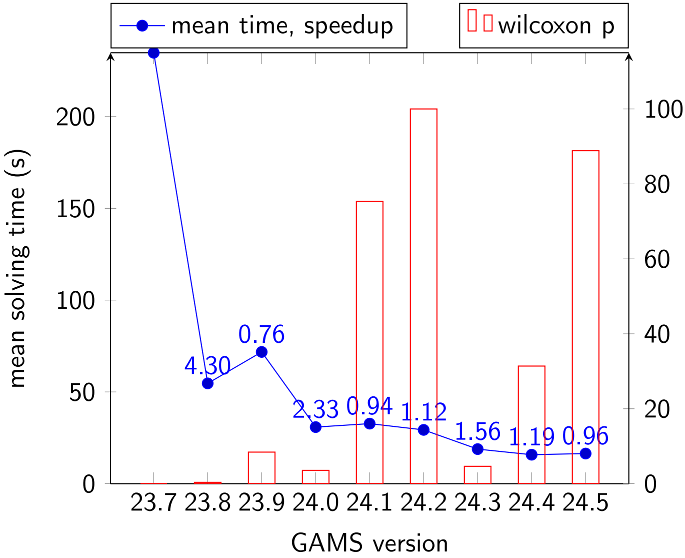
Overall speedup: 14.84

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.