Analyzing the computational impact of individual MINLP solver components

Stefan Vigerske

joint work with Ambros M. Gleixner

Zuse Institute Berlin · GAMS

MIP 2014, July 21, The Ohio State University

Outline

Analyzing MINLP solver components

Software, Hardware, Methodology

Separation

Reformulation

Primal Heuristics

Tree search

Propagation

Outline

Analyzing MINLP solver components

Software, Hardware, Methodology

Separation

Reformulation

Primal Heuristics

Tree search

Propagation

The Solver: SCIP

- a branch-cut-and-price framework
- a full-scale MIP and **MINLP solver**
- free for academic purposes, source code available, `http://scip.zib.de`

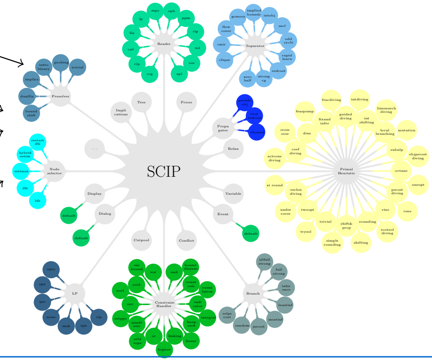
MIP
- LP relaxation
- cutting planes
- column generation

GO
- spatial branching

CP
- domain propagation

SAT
- conflict analysis
- periodic restarts

MIP, GO, CP, and SAT
- branch-and-bound



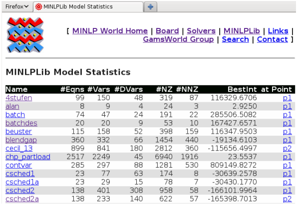
The Instances: MINLPLib 1 → 2

MINLPLib

- ▷ a collection of MINLP instances (trivial . . . challenging)
- ▷ GAMS scalar format, part of GAMS World / MINLP World

Next version (in development)

- ▷ more instances, more file formats, more statistics, . . .
- ▷ currently 822 publicly available MINLP instances
- ▷ collected from MINLPLib 1, `minlp.org`, POLIP, . . .
- ▷ see `http://www.gamsworld.org/minlp/minlplib2/html/`

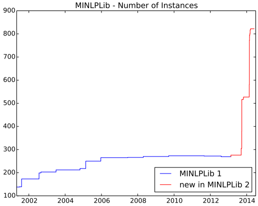


Firefox MINLPLib Model Statistics

[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



If you have interesting instances, please consider contributing.

789 choose 475

The Testset

- take MINLPLib2 α (as of April'14): 789 instances
- run SCIP with default settings
- 475 instances solved within 2 hours
- 455 instances solved within 1 hour

$\Rightarrow$ subsequent experiments: the set of 475 instances, 1 hour time limit

Hardware

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

Software

- SCIP 3.1.0.1
- SoPlex 2.0
- Ipopt 3.11.8
- CppAD 20140000.1

Averaging over heterogeneous test sets

Instances vary widely in size, nonlinearity, ...

Averaging over heterogeneous test sets

Instances vary widely in size, nonlinearity, . . . , **time to optimality**

- arithmetic average: dominated by large times
- geometric average: weights trivial and hard instances equally
- shifted geometric average: which shift?

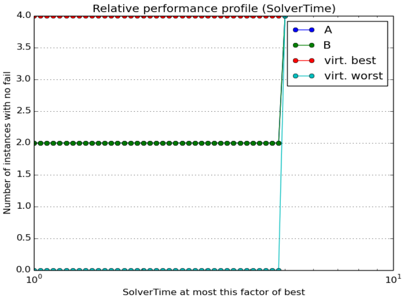
Averaging over heterogeneous test sets

Instances vary widely in size, nonlinearity, ..., time to optimality

- ▷ arithmetic average: dominated by large times
- ▷ geometric average: weights trivial and hard instances equally
- ▷ shifted geometric average: which shift?

Some results are not distinguished by performance profiles alone:

inst	A	B
1	10s	2s
2	10s	2s
3	10s	50s
4	10s	50s



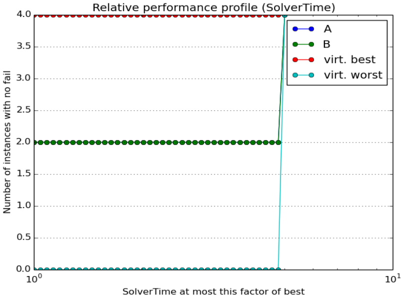
Averaging over heterogeneous test sets

Instances vary widely in size, nonlinearity, . . . , time to optimality

- arithmetic average: dominated by large times
- geometric average: weights trivial and hard instances equally
- shifted geometric average: which shift?

Some results are not distinguished by performance profiles alone:

inst	A	B
1	5x	1x
2	5x	1x
3	1x	5x
4	1x	5x



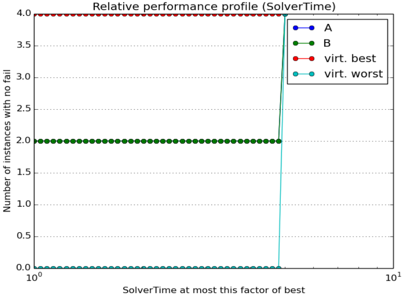
Averaging over heterogeneous test sets

Instances vary widely in size, nonlinearity, . . . , time to optimality

- ▷ arithmetic average: dominated by large times
- ▷ geometric average: weights trivial and hard instances equally
- ▷ shifted geometric average: which shift?

Some results are not distinguished by performance profiles alone:

inst	A	B
1	10s	2s
2	20s	100s
3	50s	10s
4	100s	500s



The Method: Filtered Performance Diagrams

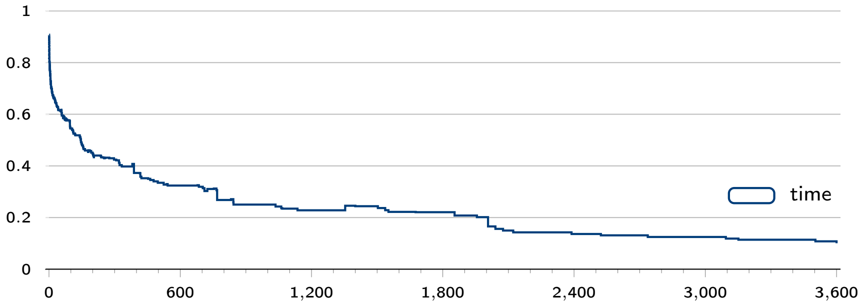
Gradually exclude instances **solved by A and B** and compute speedup:

$$t \mapsto \frac{\mu(\{t_{A,i} : \max\{t_{A,i}, t_{B,i}\} \geq t\})}{\mu(\{t_{B,i} : \max\{t_{A,i}, t_{B,i}\} \geq t\})}$$

The Method: Filtered Performance Diagrams

Gradually exclude instances **solved by A and B** and compute speedup:

$$t \mapsto \frac{\mu(\{t_{A,i} : \max\{t_{A,i}, t_{B,i}\} \geq t\})}{\mu(\{t_{B,i} : \max\{t_{A,i}, t_{B,i}\} \geq t\})}$$

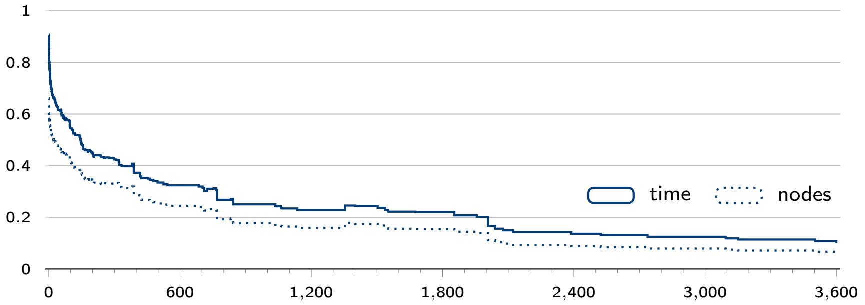


In the following: μ = geometric mean [See also Achterberg and Wunderling 2013]

The Method: Filtered Performance Diagrams

Gradually exclude instances **solved by A and B** and compute speedup:

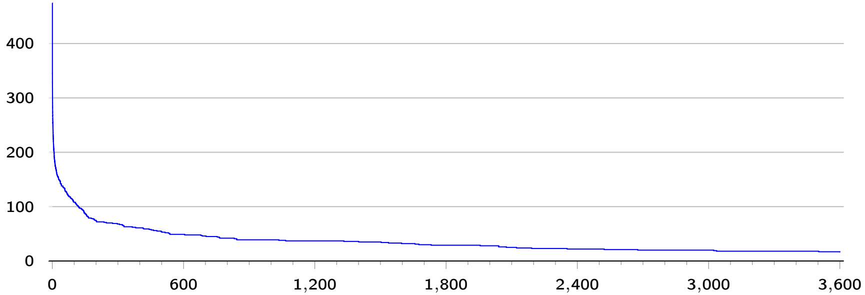
$$t \mapsto \frac{\mu(\{N_{A,i} : \max\{t_{A,i}, t_{B,i}\} \geq t\})}{\mu(\{N_{B,i} : \max\{t_{A,i}, t_{B,i}\} \geq t\})}$$



In the following: μ = geometric mean [See also Achterberg and Wunderling 2013]

Number of unsolved instances by time (default settings)

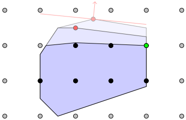
$$t \mapsto |\{i : t_i \geq t\}|$$



Outline

Analyzing MINLP solver components

Separation: MIP cutting planes

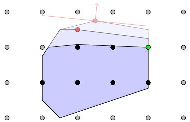


- General: Gomory, cMIR, $\{0, 1/2\}$-cuts, ...
- Problem-specific: knapsack, clique, multi commodity flow, ...

Default Settings

- run certain separators during root node
- no separation during tree search

Separation: MIP cutting planes



- General: Gomory, cMIR, $\{0, 1/2\}$-cuts, ...
- Problem-specific: knapsack, clique, multi commodity flow, ...

Default Settings

- run certain separators during root node
- no separation during tree search

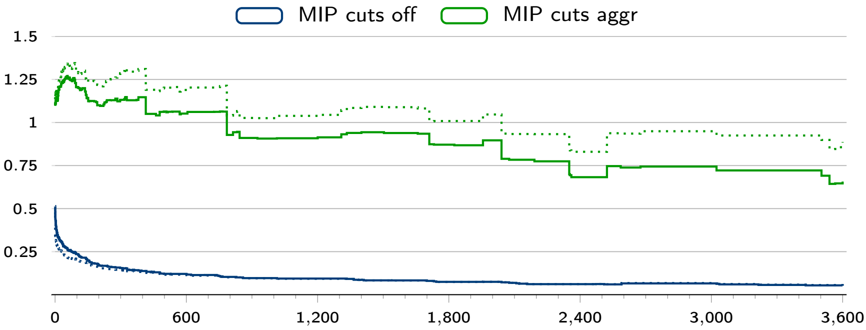
Alternative Setting I: off

Alternative Setting II: aggressive

- run separators also during tree search
- run previously disabled separators during root node

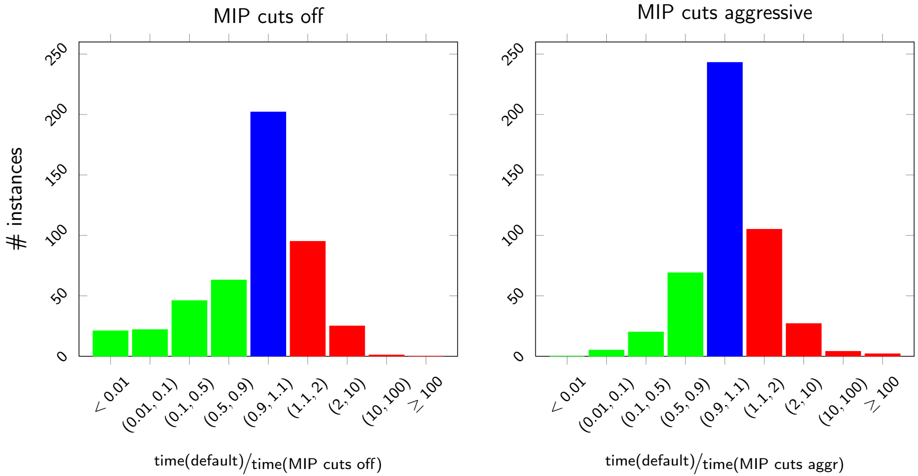
Separation: MIP cutting planes

		all		maxtime ≥ 100	
setting	solved	time	nodes	time	nodes
MIP cuts off	−39	+65%	+107%	+333%	+395%
MIP cuts aggr	−11	−7%	−10%	−18%	−23%



Separation: MIP cutting planes

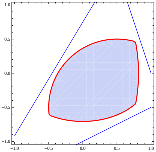
Distribution of Speedups

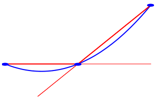


Separation: Approximation of Nonlinearities

Gradient cuts for convex terms

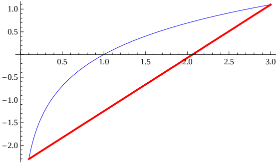
- feasibility enforced without branching
- exploit integer information for univariate convex terms

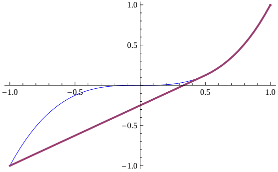


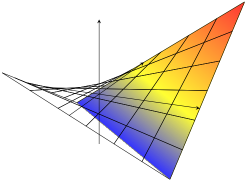


Linear underestimators for nonconvex terms

concave functions $\qquad$ $x|x|^n,\ n \geq 0$ $\qquad$ $x \cdot y$





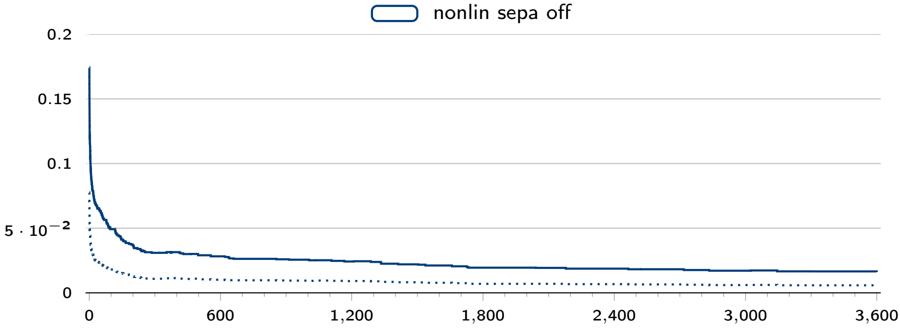


Alternative setting:

- off during fractional branching
- thus, weak relaxation of nonlinearities while branching on fractionalities

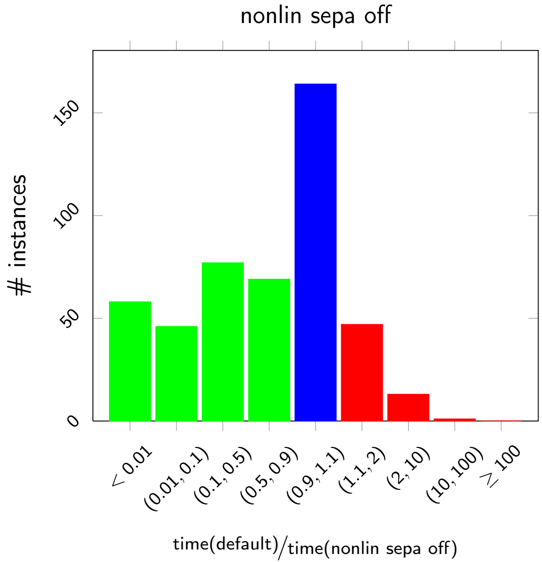
Separation: Approximation of Nonlinearities

		all		maxtime $\geq$ 100	
setting	solved	time	nodes	time	nodes
nonlin sepa off	-102	$+302\%$	$+695\%$	$+1964\%$	$+5569\%$



Separation: Approximation of Nonlinearities

Distribution of Speedups



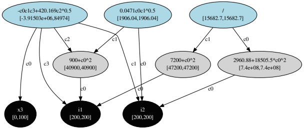
Outline

Analyzing MINLP solver components

Reformulation

Expression graph reformulation

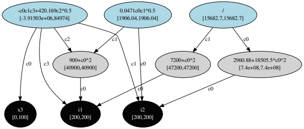
- merge expressions, e.g., polynomials
- replace subexpressions with new variables
- when switched off, only a very simple relaxation based on interval gradients is generated



Reformulation

Expression graph reformulation

- merge expressions, e.g., polynomials
- replace subexpressions with new variables
- when switched off, only a very simple relaxation based on interval gradients is generated



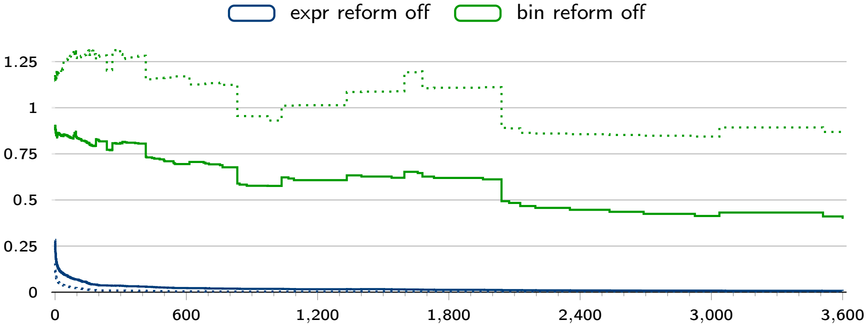
Products with binary variables

- linearize using big-M

$$x \cdot \sum_k a_k y_k \quad \text{with} \quad x \in \{0,1\}$$
$$\downarrow$$
$$M^L x \leq w \leq M^U x,$$
$$\sum_k a_k y_k - M^U(1-x) \leq w \leq \sum_k a_k y_k - M^L(1-x)$$

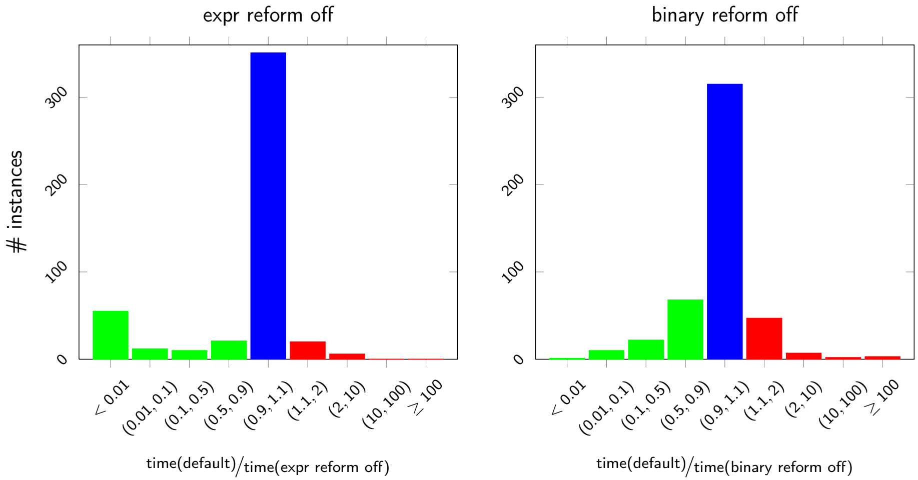
Reformulation

		all		maxtime ≥ 100	
setting	solved	time	nodes	time	nodes
expr reform off	−69	+160%	+322%	+1386%	+3631%
bin reform off	−9	+8%	−11%	+20%	−21%



Reformulation

Distribution of Speedups

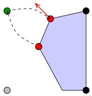


Outline

Analyzing MINLP solver components

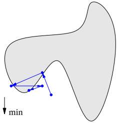
Primal Heuristics

Besides waiting for feasible LP solutions . . .



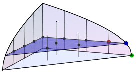
Standard MIP heuristics applied to MIP relaxation

- ▷ rounding, diving, feasibility pump, . . .
- ▷ large neighborhood search (RENS, RINS, ...)



NLP local search

- ▷ for integer and LP feasible solutions
- ▷ fix integers and solve remaining NLP (Ipopt)

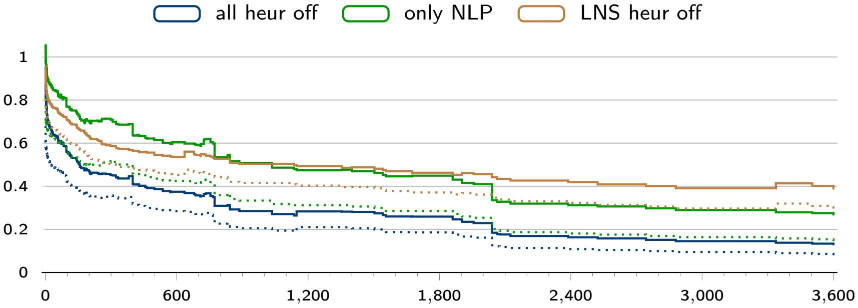


MINLP heuristics

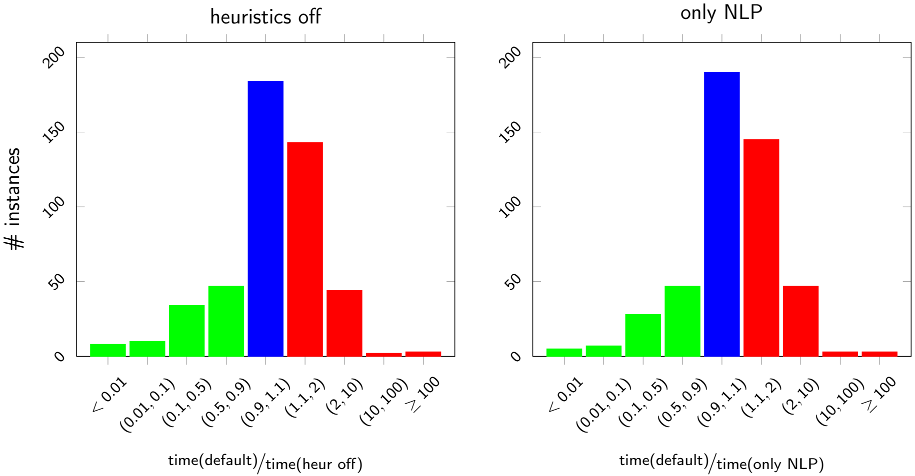
- ▷ NLP diving
- ▷ RENS [Berthold 2013]
- ▷ Undercover [Berthold and Gleixner 2013]
- ▷ . . .

Primal Heuristics

		all		maxtime ≥ 100	
setting	solved	time	nodes	time	nodes
all heur off	−19	+7%	+36%	+84%	+144%
only NLP	−11	−4%	+22%	+33%	+22%
LNS heur off	−10	+4%	+20%	+51%	+71%

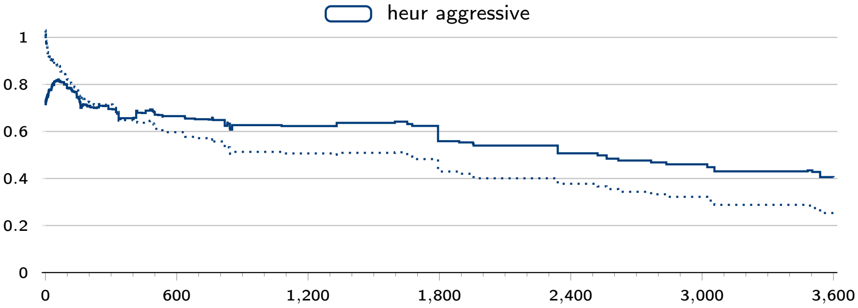


Distribution of Speedups



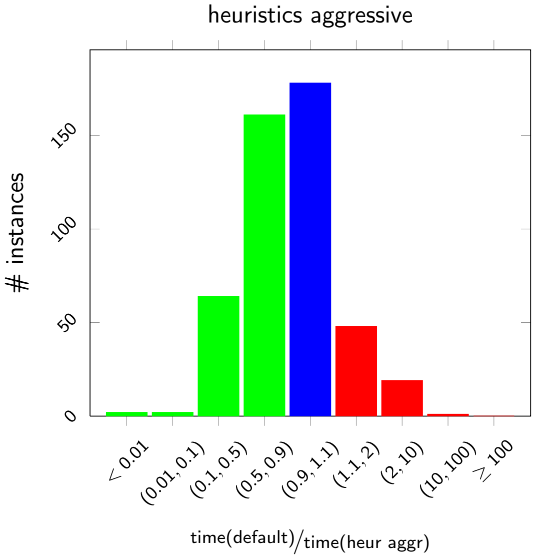
Primal Heuristics

		all		maxtime ≥ 100	
setting	solved	time	nodes	time	nodes
heur aggressive	−2	+27%	−4%	+28%	+86%



Primal Heuristics

Distribution of Speedups



Outline

Analyzing MINLP solver components

Software, Hardware, Methodology

Separation

Reformulation

Primal Heuristics

Tree search

Propagation

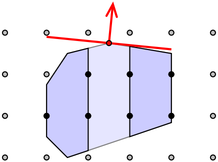
Branching

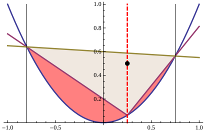
Reliability (MIP) | Inference[1] (CP) | VSIDS[2] (SAT)

1. on "fractional" integer vars

2. on vars in violated nonlinear terms

pseudo-cost (GO)





[See Tawarmalani and Sahinidis 2002, Achterberg and Berthold 2009, Belotti et al. 2009, . . .]

[1] Inference branching: prefer variables where branching resulted in high number of domain propagation before

[2] VSIDS: prefer variables used to produce recent conflict constraints

Branching

Reliability (MIP) | Inference[1] (CP) | VSIDS[2] (SAT)

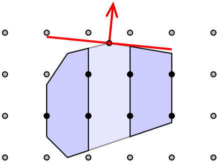
↓

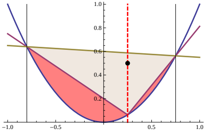
1. on “fractional” integer vars

2. on vars in violated nonlinear terms

↑

pseudo-cost (GO)





Alternative settings for spatial branching

- inference[1], most infeasible, random

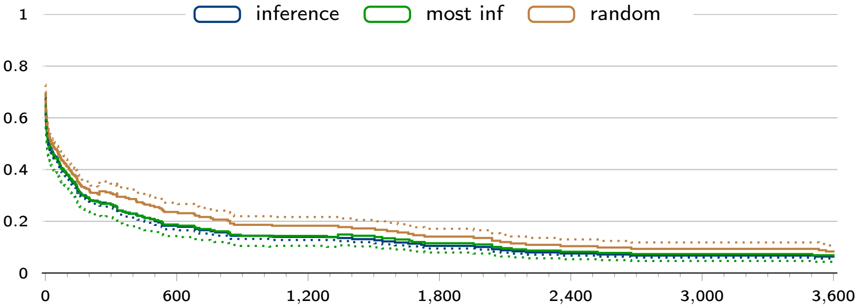
[See Tawarmalani and Sahinidis 2002, Achterberg and Berthold 2009, Belotti et al. 2009, . . .]

[1] Inference branching: prefer variables where branching resulted in high number of domain propagation before

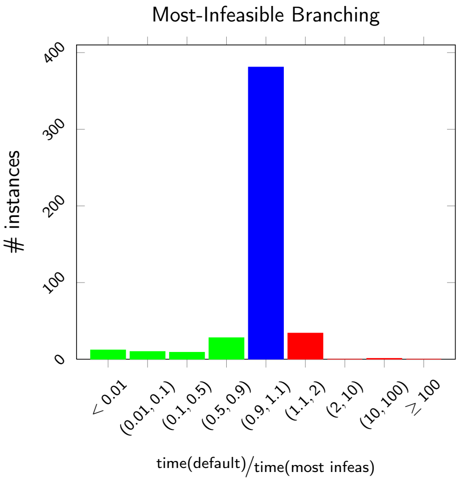
[2] VSIDS: prefer variables used to produce recent conflict constraints

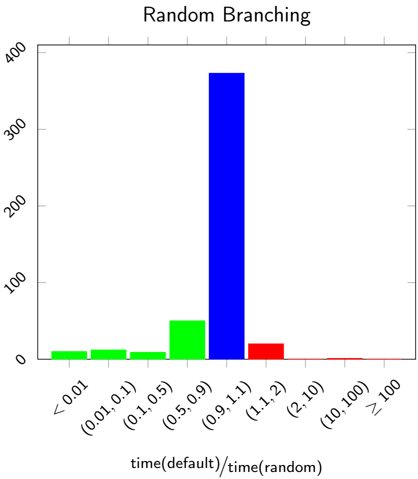
Spatial Branching

		all		maxtime $\geq$ 100	
setting	solved	time	nodes	time	nodes
inference	-27	$+31\%$	$+34\%$	$+167\%$	$+176\%$
most inf	-24	$+30\%$	$+38\%$	$+165\%$	$+209\%$
random	-24	$+30\%$	$+28\%$	$+145\%$	$+130\%$



Distribution of Speedups

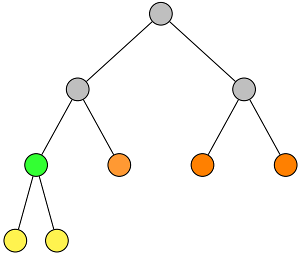




Node selection

Tasks

- improve primal bound
- keep computational effort small
- improve global dual bound



Best estimate with plunging

- select node Q with best/minimal (pseudo cost) estimate value for feasible solution objective value

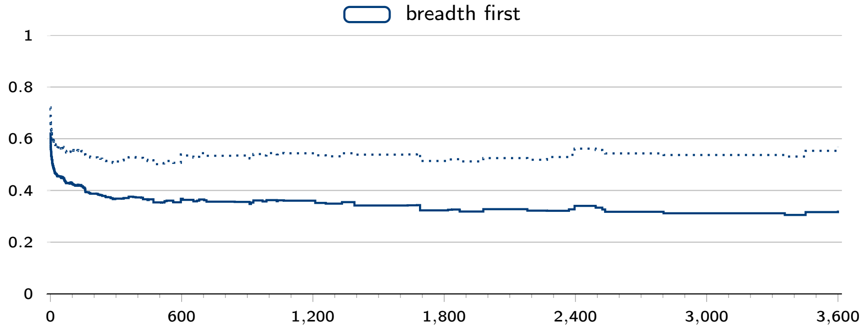
$$\bar{z}_Q + \sum_{k:\bar{x}_k \text{fractional}} \min\{\Psi^- f^-, \Psi^+ f^+\}$$

- plunge (diving with single backtrack)

Alternative setting: breadth first search

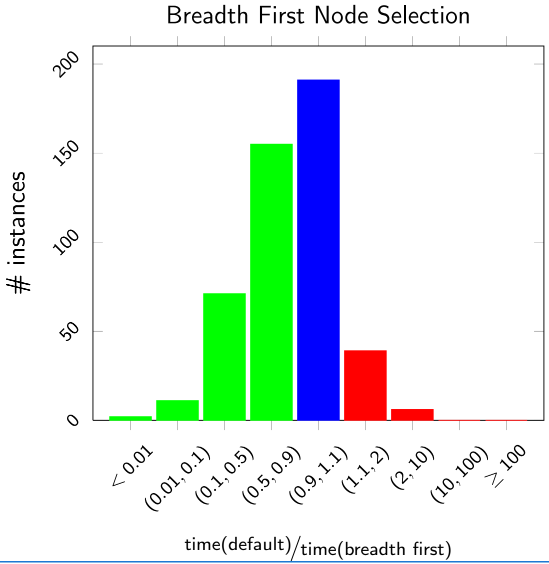
Node selection

		all		maxtime ≥ 100	
setting	solved	time	nodes	time	nodes
breadth first	−22	+42%	+29%	+136%	+81%



Node Selection

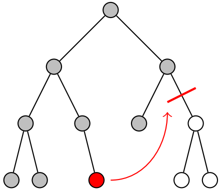
Distribution of Speedups



Conflict analysis / “nogood” learning

Analyse reason for pruning a node

- branchings and propagations
- infeasible and bound exceeding LP relaxation: dual ray heuristic
- derive short nogoods/conflict constraints
- most nonlinear constraints do not participate in conflict analysis yet



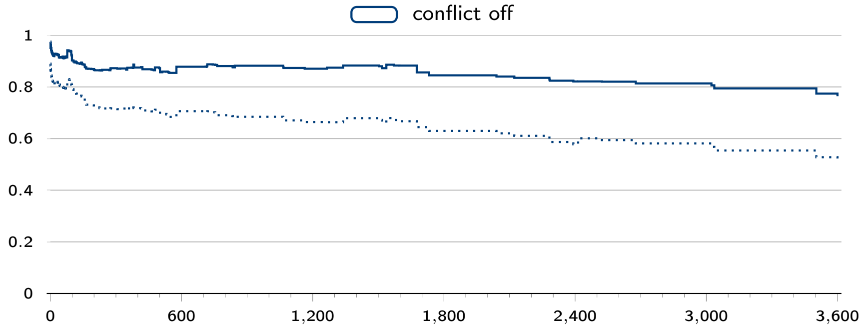
$$x_1 - x_3 \leq 0$$

Use subsequently

- to cut off other nodes
- to enable further propagations
- for VSIDS in branching

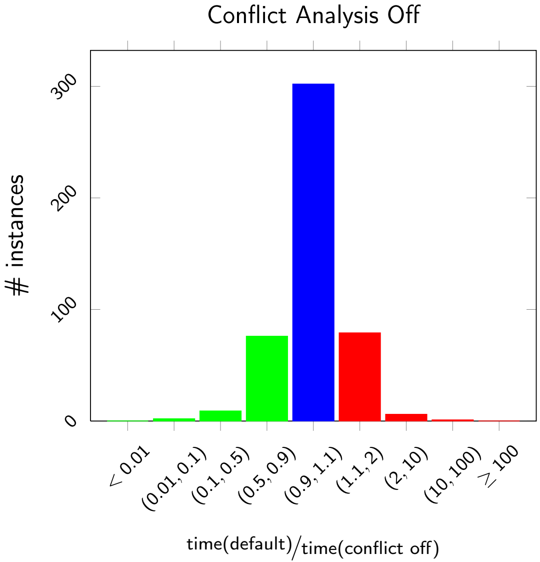
Conflict analysis / “nogood” learning

		all		maxtime $\geq$ 100	
setting	solved	time	nodes	time	nodes
conflict off	-2	$+2\%$	$+9\%$	$+11\%$	$+27\%$



Conflict Analysis / "nogood" learning

Distribution of Speedups



Outline

Analyzing MINLP solver components

Software, Hardware, Methodology

Separation

Reformulation

Primal Heuristics

Tree search

Propagation

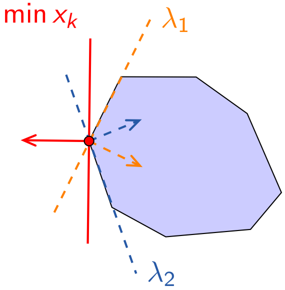
Bound tightening/propagation

Particularly important for nonconvex MINLP

- branching on continuous variables/infinite domains
- tight domains $\rightsquigarrow$ tight relaxation

Primal and dual reductions

- reduced cost
- probing on binaries
- FBBT: feasibility-based bound tightening
- OBBT: optimization-based bound tightening and Lagrangian variable bounds:



$$x_k \geq \sum_{i:r_i>0} r_i \, \underline{x}_i + \sum_{i:r_i<0} r_i \, \overline{x}_i + \mu \, c^T x^* + \lambda^T b$$

[Ryoo and Sahinidis 1996, Belotti et al. 2009, Gleixner and Weltge 2013, ...]

Propagating Lagrangian Variable Bounds (LVBs)

The right-hand side of $x_k \geq \underline{r}^T \underline{x} + \overline{r}^T \overline{x} + \mu c^T x^* + \lambda^T b$ is tightened

- if some variable lower bound $\underline{x}_i$ increases for $\underline{r}_i > 0$
- if some variable upper bound $\overline{x}_i$ decreases for $\overline{r}_i < 0$
- if a better primal solution x^* is found and $\mu < 0$

[Gleixner and Weltge 2013]

Propagating Lagrangian Variable Bounds (LVBs)

The right-hand side of $x_k \geq \underline{r}^T \underline{x} + \overline{r}^T \overline{x} + \mu c^T x^* + \lambda^T b$ is tightened

- if some variable lower bound $\underline{x}_i$ increases for $\underline{r}_i > 0$
- if some variable upper bound $\overline{x}_i$ decreases for $\overline{r}_i < 0$
- if a better primal solution x^* is found and $\mu < 0$

Learn LVBs during root OBBT and propagate again

- locally at nodes of the branch-and-bound tree
- globally if a better primal solution is found
- compare "duality-based reduction" [Tawarmalani and Sahinidis 2004]

[Gleixner and Weltge 2013]

Propagating Lagrangian Variable Bounds (LVBs)

The right-hand side of $x_k \geq \underline{r}^T \underline{x} + \overline{r}^T \overline{x} + \mu c^T x^* + \lambda^T b$ is tightened

- if some variable lower bound $\underline{x}_i$ increases for $\underline{r}_i > 0$
- if some variable upper bound $\overline{x}_i$ decreases for $\overline{r}_i < 0$
- if a better primal solution x^* is found and $\mu < 0$

Learn LVBs during root OBBT and propagate again

- locally at nodes of the branch-and-bound tree
- globally if a better primal solution is found
- compare "duality-based reduction" [Tawarmalani and Sahinidis 2004]

Computational Experience

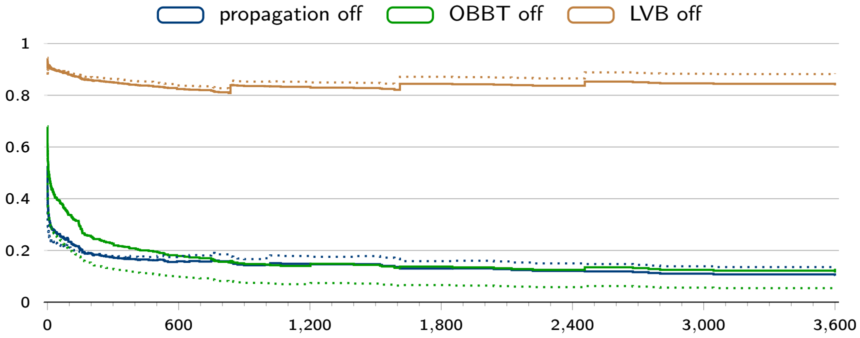
- on every other MINLP, at least one nontrivial LVB from every 2nd OBBT LP
- LVB propagation typically $\leq 2\%$ of total running time, when implemented efficiently

This promises a computationally cheap approximation of OBBT in the tree.

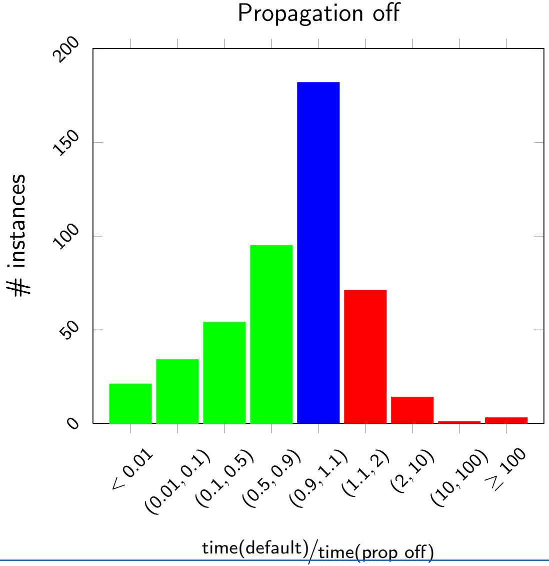
[Gleixner and Weltge 2013]

Bound tightening/propagation

setting	solved	all time	all nodes	maxtime $\geq$ 100 time	maxtime $\geq$ 100 nodes
propagation off	−48	+90%	+129%	+332%	+378%
OBBT off	−25	+47%	+93%	+198%	+396%
LVB off	−4	+6%	+9%	+18%	+17%

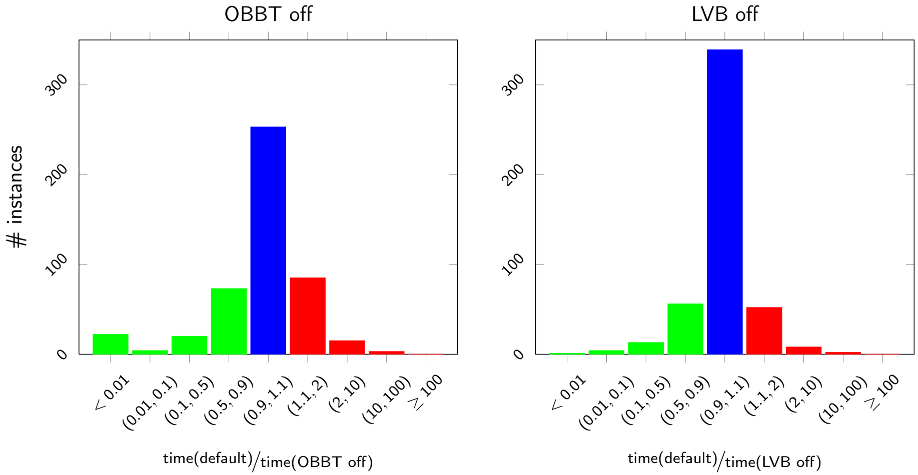


Distribution of Speedups



Bound tightening/propagation

Distribution of Speedups



Summary

		all		maxtime $\geq$ 100	
setting	solved	time	nodes	time	nodes
nonlin sepa off	−102	+302%	+695%	+1964%	+5569%
expr reform off	−69	+160%	+322%	+1386%	+3631%
propagation off	−48	+90%	+129%	+397%	+461%
MIP cuts off	−39	+65%	+107%	+333%	+395%
inference branching	−27	+31%	+34%	+167%	+176%
OBBT off	−25	+47%	+93%	+303%	+607%
most inf branching	−24	+30%	+38%	+165%	+209%
random branching	−24	+30%	+28%	+145%	+130%
breadth first search	−22	+42%	+29%	+136%	+81%
all heur off	−19	+7%	+36%	+84%	+144%
MIP cuts aggr	−11	−7%	−10%	−18%	−23%
only NLP heur	−11	−4%	+22%	+33%	+22%
LNS heur off	−10	+4%	+20%	+51%	+71%
bin reform off	−9	+8%	−11%	+20%	−21%
LVB off	−4	+6%	+9%	+20%	+19%
heur aggressive	−2	+27%	−4%	+28%	+86%
conflict off	−2	+2%	+9%	+11%	+27%