Performance of COIN-OR solvers for the solution of MINLPs using GAMS

Michael Bussieck
Jan-Hendrik Jagla

Stefan Vigerske

22nd European Conference on Operational Research
Prague, July 10, 2007

Introduction

COIN-OR = Common Infrastructor for Operations Research (www.coin-or.org)

- initiative to spur the development of open-source software for the operations research community
- COIN-OR sessions TD-48, TE-48 (today 15:00-18:30, room RB 213)
- currently 28 projects in different categories:
  - developer tools, utilities
  - interfaces
  - modeling
  - solvers for LP, MIP, NLP, MINLP, SDP, ...
  - ...

Introduction

COIN-OR = Common Infrastructor for Operations Research (www.coin-or.org)

- initiative to spur the development of open-source software for the operations research community
- COIN-OR sessions TD-48, TE-48 (today 15:00-18:30, room RB 213)
- currently 28 projects in different categories:
  - developer tools, utilities
  - interfaces
  - modeling
  - solvers for LP, MIP, NLP, MINLP, SDP, ...
  - ...
- GAMSlinks project: development of links between GAMS and open source solvers

Introduction

COIN-OR = Common Infrastructor for Operations Research (www.coin-or.org)

- initiative to spur the development of open-source software for the operations research community
- COIN-OR sessions TD-48, TE-48 (today 15:00-18:30, room RB 213)
- currently 28 projects in different categories:
    - developer tools, utilities
    - interfaces
    - modeling
    - solvers for LP, MIP, NLP, MINLP, SDP, ...
    - ...
- GAMSlinks project: development of links between GAMS and open source solvers

In this talk: Performance of COIN-OR solvers for the solution of MINLPs, when used as NLP subsolver, MIP subsolver, or MINLP solver.

Overview

MINLPLib

- part of the MINLP World at `www.gamsworld.org`
- collection of 260 Mixed Integer Nonlinear Programming models
- both theoretical and practical test models

MINLPLib

- part of the MINLP World at `www.gamsworld.org`
- collection of 260 Mixed Integer Nonlinear Programming models
- both theoretical and practical test models
- we selected models with at most 1000 variables

$\Rightarrow$ 210 models:

- 86 are convex
- 124 are nonconvex

The MINLP solvers (1/2)

SBB = Simple Branch and Bound

- developed by A. Drud (ARKI Consulting & Development A/S) and M. Bussieck (GAMS)
- lower bounds: solving NLP = MINLP with relaxed integrality requ.
- upper bounds: from NLP solution, if integer-feasible
- heuristic, if model is not convex
- can choose any GAMS NLP solver as subsolver
- no LP/MIP solver required

The MINLP solvers (1/2)

SBB = Simple Branch and Bound

- developed by A. Drud (ARKI Consulting & Development A/S) and M. Bussieck (GAMS)
- lower bounds: solving NLP = MINLP with relaxed integrality requ.
- upper bounds: from NLP solution, if integer-feasible
- heuristic, if model is not convex
- can choose any GAMS NLP solver as subsolver
- no LP/MIP solver required

DICOPT = Discrete and Continuous Optimizer

- developed by J. Viswanathan and I.E. Grossmann at CMU
- outer approximation by linearization ⇒ MIP master problem
- subproblems by fixing discrete variables ⇒ NLP subproblems
- alternating solving of MIP and NLP until NLP solution is worsening
- heuristic, if model is not convex
- can choose any GAMS NLP and MIP solver as subsolver

The MINLP solvers (2/2)

αECP 1.30 = α Extended Cutting Plane

- developed by T. Westerlund and T. Lastusilta at Åbo Akademi University, Finland
- outer approximation by linearization $\Rightarrow$ MIP
- shifting of hyperplanes in case of infeasible MIP
- upper bound from MINLP-feasible MIP solutions
- heuristic, if model is not pseudo-convex
- can choose any GAMS MIP solver as subsolver

The MINLP solvers (2/2)

αECP 1.30 = α Extended Cutting Plane

- developed by T. Westerlund and T. Lastusilta at Åbo Akademi University, Finland
- outer approximation by linearization $\Rightarrow$ MIP
- shifting of hyperplanes in case of infeasible MIP
- upper bound from MINLP-feasible MIP solutions
- heuristic, if model is not pseudo-convex
- can choose any GAMS MIP solver as subsolver

BONMIN = Basic Open-source Nonlinear Mixed Integer programming

- COIN-OR solver developed by P. Bonami et.al. at CMU and IBM
- outer-approximation based branch-and-cut algorithm
- lower bounds: solving LP relaxation (COIN-OR CLP)
- upper bounds: solve MINLP with fixed discrete variables (IPOPT)
- heuristic, if model is not convex
- cannot choose GAMS NLP or LP subsolver

The NLP solvers

CONOPT 3.14r

- developed by A. Drud at ARKI Consulting & Development A/S
- SLP and SQP algorithms
- default GAMS NLP solver

The NLP solvers

CONOPT 3.14r

- developed by A. Drud at ARKI Consulting & Development A/S
- SLP and SQP algorithms
- default GAMS NLP solver

KNITRO 5.1

- developed by R.A. Waltz at Ziena Optimization, Inc.
- interior-point and active-set SLQP algorithms

The NLP solvers

CONOPT 3.14r

- developed by A. Drud at ARKI Consulting & Development A/S
- SLP and SQP algorithms
- default GAMS NLP solver

KNITRO 5.1

- developed by R.A. Waltz at Ziena Optimization, Inc.
- interior-point and active-set SLQP algorithms

IPOPT 3.3 = Interior Point Optimizer

- COIN-OR solver developed by A. Wächter at IBM
- interior-point algorithm

The MIP/LP solvers

CPLEX 10.20

- developed by ILOG, Inc.
- primal/dual simplex and barrier optimizer for LPs
- branch-and-cut for MIPs

The MIP/LP solvers

CPLEX 10.20

- developed by ILOG, Inc.
- primal/dual simplex and barrier optimizer for LPs
- branch-and-cut for MIPs

CLP (June 2006) = COIN-OR Linear Programming
CBC (June 2006) = COIN-OR Branch-and-Cut

- COIN-OR solver developed by J. Forrest at IBM
- primal/dual simplex and barrier optimizer for LPs
- branch-and-cut for MIPs

Overview

Performance Tools and GAMS Bench Solver

Performance World (at `www.gamsworld.org`):

- PerformanceLib: Libraries of test problems
- Performance Tools: simplifying performance data collection, measurement, postprocessing, and visualization
- PAVER: Server for Automated Performance Analysis & Visualization

GAMS BENCH solver:

- facilitate benchmarking of GAMS optimization solvers
- calls user-specified GAMS solvers for particular modeltype
- captures results in list file
- can call GAMS/EXAMINER solver to verify feasibility and optimality of returned solution

Using Performance Tools and Bench solver

1. Create a list of models ⇒ `MINLPs.list`

Using Performance Tools and Bench solver

1. Create a list of models ⇒ `MINLPs.list`
2. Choose solvers by creating a BENCH option file ⇒ `bench.opt`

```
solvers sbb.1 sbb.2 sbb.3 dicopt.1 dicopt.2 dicopt.3 ...
paver 1
```

Using Performance Tools and Bench solver

1. Create a list of models ⇒ `MINLPs.list`
2. Choose solvers by creating a BENCH option file ⇒ `bench.opt`

```
solvers sbb.1 sbb.2 sbb.3 dicopt.1 dicopt.2 dicopt.3 ...
paver 1
```

3. Create a batch file using `crbatch.gms` ⇒ `MINLPs_batch.gms`:

```
gams crbatch.gms --batfile=MINLPs_batch --modelfile=MINLPs.list
 --type=MINLP --solver=bench --optfile=1 --reslim=3600 --optcr=0.01
```

Using Performance Tools and Bench solver

1. Create a list of models ⇒ `MINLPs.list`
2. Choose solvers by creating a BENCH option file ⇒ `bench.opt`
   ```
   solvers sbb.1 sbb.2 sbb.3 dicopt.1 dicopt.2 dicopt.3 ...
   paver 1
   ```
3. Create a batch file using `crbatch.gms` ⇒ `MINLPs_batch.gms`:
   ```
   gams crbatch.gms --batfile=MINLPs_batch --modelfile=MINLPs.list
    --type=MINLP --solver=bench --optfile=1 --reslim=3600 --optcr=0.01
   ```
4. Start benchmarks to create trace files ⇒ `sbb-1.pvr`, `sbb-2.pvr`, ...
   ```
   gams MINLPs_batch.gms
   ```

Using Performance Tools and Bench solver

1. Create a list of models ⇒ `MINLPs.list`
2. Choose solvers by creating a BENCH option file ⇒ `bench.opt`

   ```
   solvers sbb.1 sbb.2 sbb.3 dicopt.1 dicopt.2 dicopt.3 ...
   paver 1
   ```

3. Create a batch file using `crbatch.gms` ⇒ `MINLPs_batch.gms`:

   ```
   gams crbatch.gms --batfile=MINLPs_batch --modelfile=MINLPs.list
    --type=MINLP --solver=bench --optfile=1 --reslim=3600 --optcr=0.01
   ```

4. Start benchmarks to create trace files ⇒ `sbb-1.pvr`, `sbb-2.pvr`, ...

   ```
   gams MINLPs_batch.gms
   ```

5. Compare solvers, generate performance profiles using `pprocess.gms`:

   ```
   gams pprocess --trace1=sbb-1.pvr --trace2=sbb-2.pvr ...
   ```

   (or let the PAVER server do this job)

Using Performance Tools and Bench solver

1. Create a list of models ⇒ `MINLPs.list`
2. Choose solvers by creating a BENCH option file ⇒ `bench.opt`
   ```
   solvers sbb.1 sbb.2 sbb.3 dicopt.1 dicopt.2 dicopt.3 ...
   paver 1
   ```
3. Create a batch file using `crbatch.gms` ⇒ `MINLPs_batch.gms`:
   ```
   gams crbatch.gms --batfile=MINLPs_batch --modelfile=MINLPs.list
    --type=MINLP --solver=bench --optfile=1 --reslim=3600 --optcr=0.01
   ```
4. Start benchmarks to create trace files ⇒ `sbb-1.pvr`, `sbb-2.pvr`, ...
   ```
   gams MINLPs_batch.gms
   ```
5. Compare solvers, generate performance profiles using `pprocess.gms`:
   ```
   gams pprocess --trace1=sbb-1.pvr --trace2=sbb-2.pvr ...
   ```
   (or let the PAVER server do this job)

⇒ performance profiles, comparision of solver outcomes, comparision of solver resource times

Overview

The Models and the Solvers

Benchmarking with GAMS

- Performance Tools and GAMS Bench solver
- Performance Profiles

Performance of MINLP solvers on MINLPLib

- Performance of NLP subsolvers
- Performance of MIP subsolvers

Conclusions

Performance Profiles

E.D. Dolan and J.J. More, Mathematical Programming, 91, 2002:

- compare performance of solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ with best performance by any solver on problem p:

$$\rho(p, s) := \frac{t_{p,s}}{\min_{s' \in \mathcal{S}} t_{p,s'}}$$

- $t_{p,s}$ = time solver s spend on p, $\quad t_{p,s} = \infty$ if s did not *solve* p

Performance Profiles

E.D. Dolan and J.J. More, Mathematical Programming, 91, 2002:

- compare performance of solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ with best performance by any solver on problem p:

$$\rho(p, s) := \frac{t_{p,s}}{\min_{s' \in \mathcal{S}} t_{p,s'}}$$

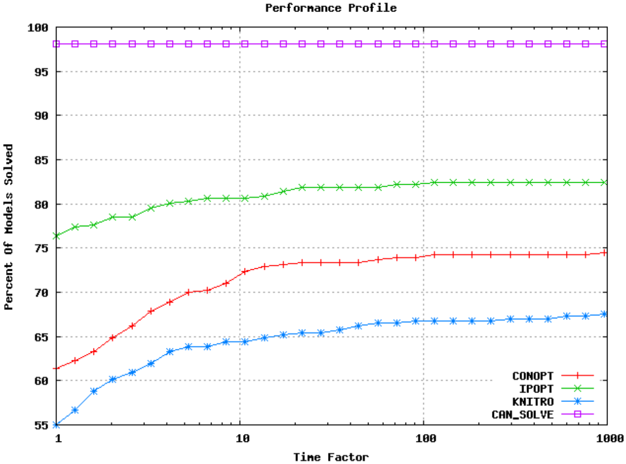
- $t_{p,s}$ = time solver s spend on p, $\quad t_{p,s} = \infty$ if s did not *solve* p
- $P_s(\tau)$ = probability that performance ratio $\rho(p, s)$ within factor of τ of best possible ratio:

$$P_s(\tau) := \frac{|\{p \in \mathcal{P} : \rho(p, s) \leq \tau\}|}{|\mathcal{P}|}$$

- percentage of models that solver s will *solve* if for each model, s can have a maximum resource time of τ times the minimum time
- s solved p: found feasible point *or* found best solution among all solvers

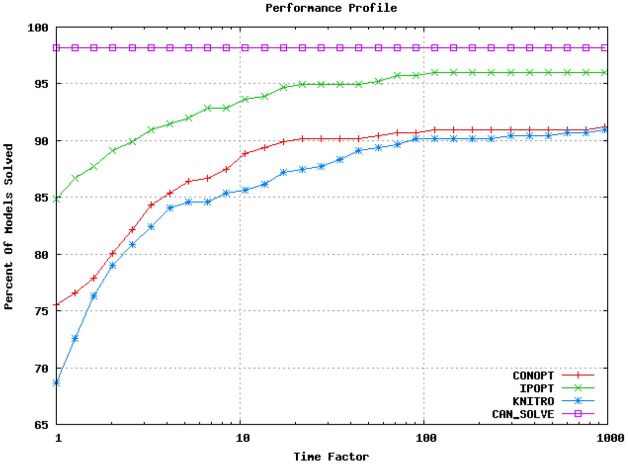
Example: Performance Profiles for NLP solvers

- NLP solvers on GlobalLib (379 models), timelimit: 1 hour
- solved = found best solution among all solvers



Example: Performance Profiles for NLP solvers

- NLP solvers on GlobalLib (379 models), timelimit: 1 hour
- solved = found some feasible point



Overview

The Models and the Solvers

Benchmarking with GAMS

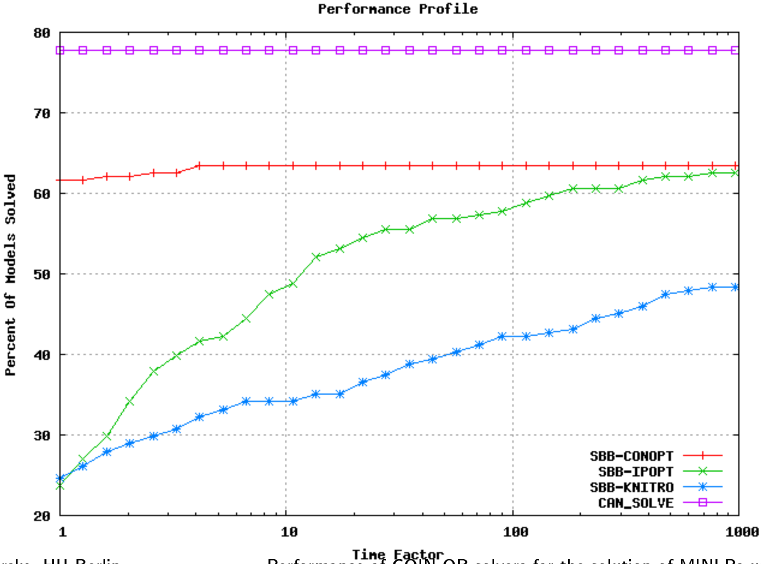
- Performance Tools and GAMS Bench solver
- Performance Profiles

Performance of MINLP solvers on MINLPLib

- Performance of NLP subsolvers
- Performance of MIP subsolvers

Conclusions

SBB on all MINLPs, different NLP solver



Improving robustness of SBB

- SBB option `failseq`: sequence of NLP solvers for resolving a subproblem when first solver fails
- SBB option `infeasseq`: sequence of NLP solvers for resolving a subproblem when first solver returns infeasible and SBB is high in the search tree

$\Rightarrow$ increase robustness against failures of NLP solver and nonconvexities

Improving robustness of SBB

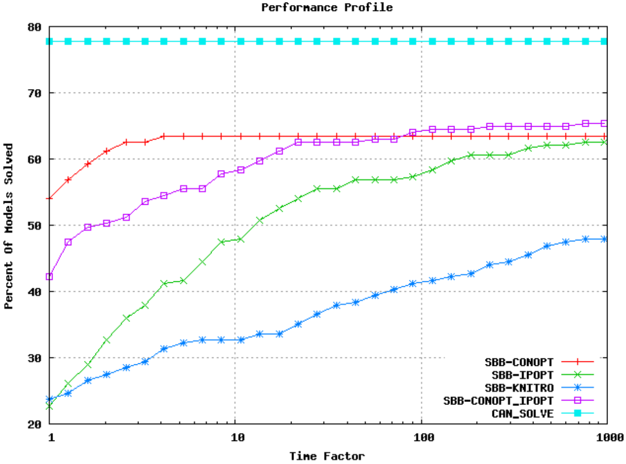
- SBB option `failseq`: sequence of NLP solvers for resolving a subproblem when first solver fails
- SBB option `infeasseq`: sequence of NLP solvers for resolving a subproblem when first solver returns infeasible and SBB is high in the search tree

$\Rightarrow$ increase robustness against failures of NLP solver and nonconvexities

$\Rightarrow$ let SBB call IPOPT if CONOPT fails or reports infeasible and SBB is at most at depth 30 of the search tree

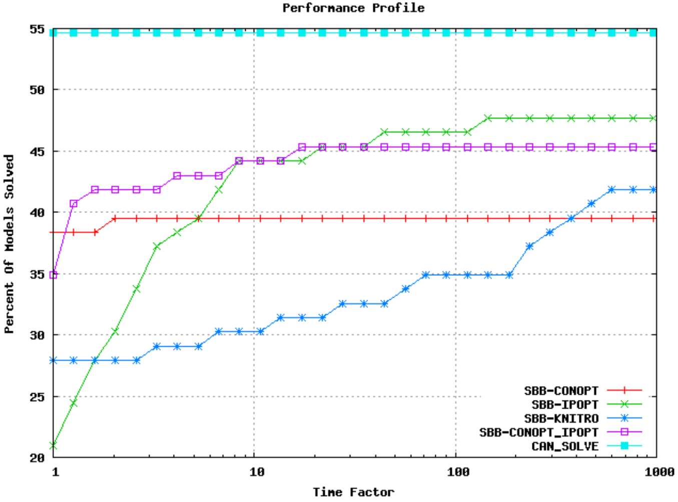
SBB with failseq and infeasseq

⇒ let SBB call IPOPT if CONOPT fails or reports infeasible and SBB is at most at depth 30 of the search tree

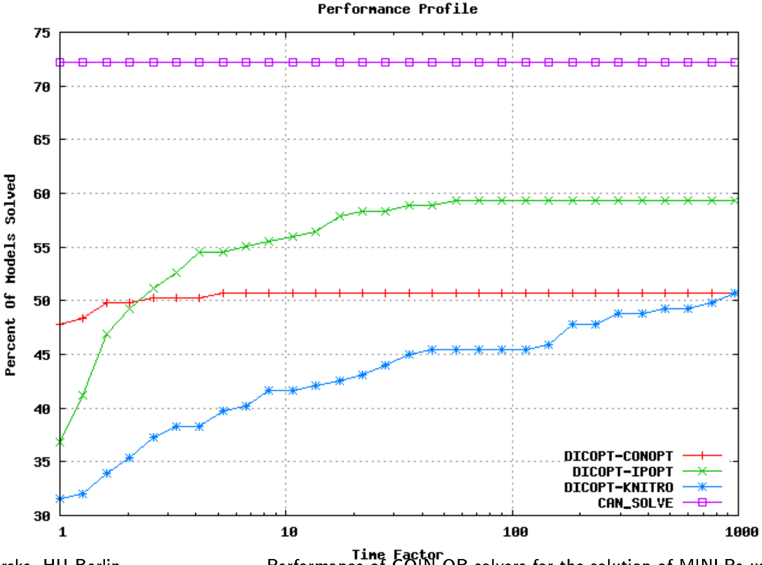


SBB on convex MINLPs

consider only 86 of the 215 models which are convex

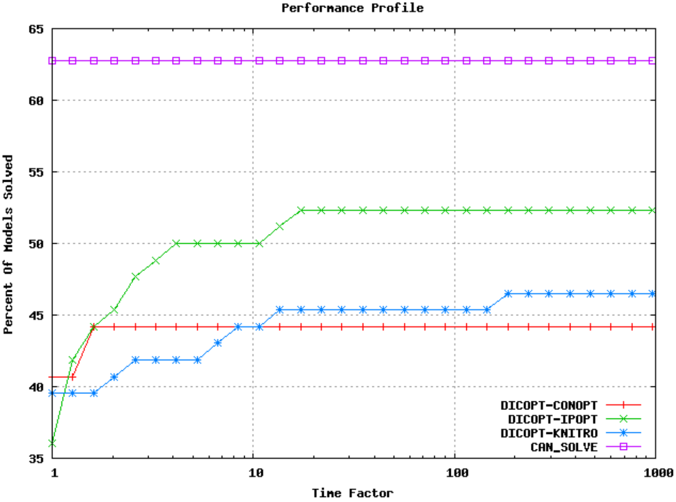


DICOPT on all MINLPs, different NLP solver



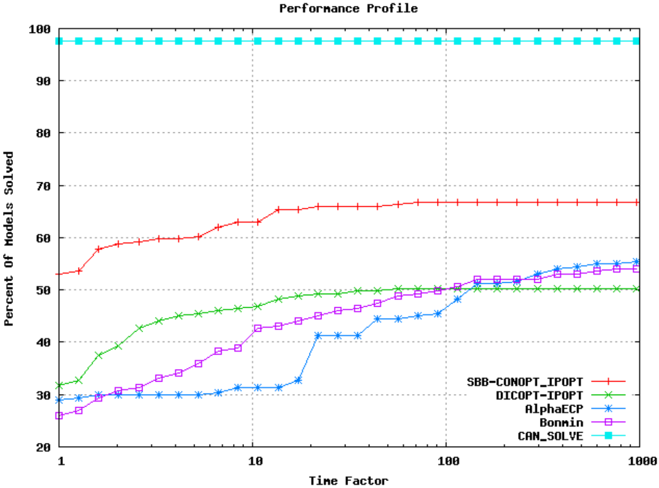
DICOPT on convex MINLPs

consider only 86 of the 215 models which are convex



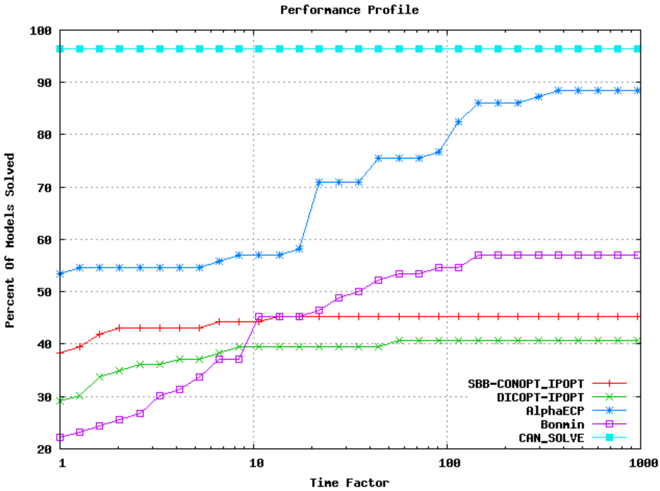
SBB vs. DICOPT vs. AlphaECP vs. Bonmin

- SBB with CONOPT and IPOPT as subsolvers
- DICOPT with IPOPT as subsolver



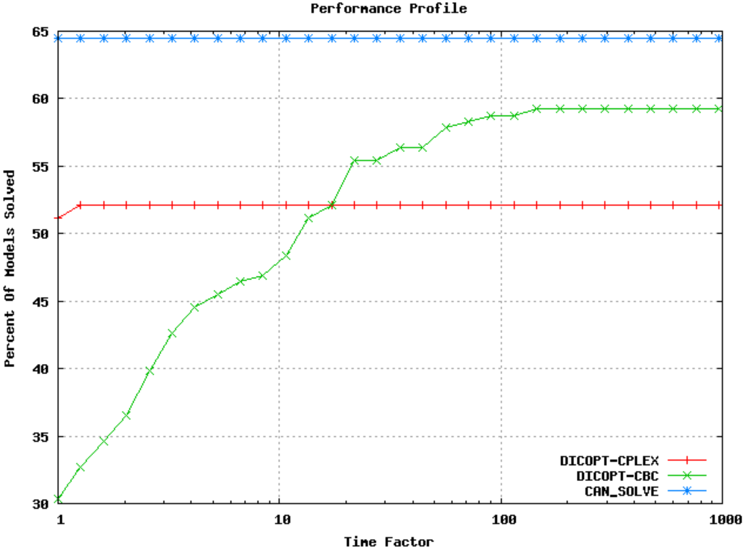
SBB, DICOPT, AlphaECP, Bonmin on convex MINLPs

- consider only 86 of the 215 models which are convex
- SBB with CONOPT and IPOPT; DICOPT with IPOPT

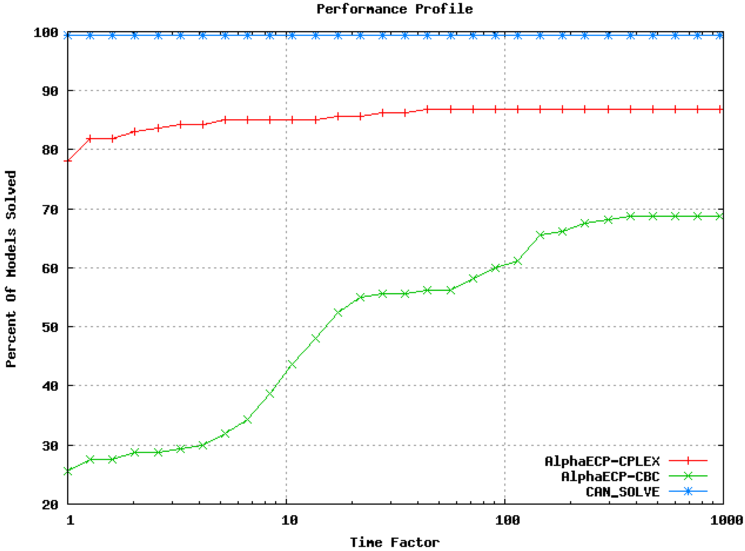


Overview

DICOPT on all MINLPs, different MIP solver



AlphaECP on all MINLPs, different MIP solver



Conclusions

all MINLPs:

- SBB with CONOPT faster than with IPOPT, same solution quality
- SBB with CONOPT+IPOPT slower than with CONOPT alone, but a few more problems solved

Conclusions

all MINLPs:

- SBB with CONOPT faster than with IPOPT, same solution quality
- SBB with CONOPT+IPOPT slower than with CONOPT alone, but a few more problems solved
- DICOPT with IPOPT solves more then CONOPT, atmost double time

Conclusions

all MINLPs:

- SBB with CONOPT faster than with IPOPT, same solution quality
- SBB with CONOPT+IPOPT slower than with CONOPT alone, but a few more problems solved
- DICOPT with IPOPT solves more then CONOPT, atmost double time
- SBB superior to other MINLP solver w.r.t. solution quality and time

Conclusions

all MINLPs:

- SBB with CONOPT faster than with IPOPT, same solution quality
- SBB with CONOPT+IPOPT slower than with CONOPT alone, but a few more problems solved
- DICOPT with IPOPT solves more then CONOPT, atmost double time
- SBB superior to other MINLP solver w.r.t. solution quality and time
- DICOPT with CBC slower than with CPLEX, but more problems solved
- AlphaECP with CPLEX better than with CBC w.r.t. time and solution quality

Conclusions

all MINLPs:

- SBB with CONOPT faster than with IPOPT, same solution quality
- SBB with CONOPT+IPOPT slower than with CONOPT alone, but a few more problems solved
- DICOPT with IPOPT solves more then CONOPT, atmost double time
- SBB superior to other MINLP solver w.r.t. solution quality and time
- DICOPT with CBC slower than with CPLEX, but more problems solved
- AlphaECP with CPLEX better than with CBC w.r.t. time and solution quality

only convex MINLPs:

- SBB or DICOPT with IPOPT or KNITRO solves more than with CONOPT
- AlphaECP superior to Bonmin superior to SBB and DICOPT w.r.t. solution qual.
- AlphaECP superior to SBB superior to Bonmin and DICOPT w.r.t. time

Conclusions

all MINLPs:

- SBB with CONOPT faster than with IPOPT, same solution quality
- SBB with CONOPT+IPOPT slower than with CONOPT alone, but a few more problems solved
- DICOPT with IPOPT solves more then CONOPT, atmost double time
- SBB superior to other MINLP solver w.r.t. solution quality and time
- DICOPT with CBC slower than with CPLEX, but more problems solved
- AlphaECP with CPLEX better than with CBC w.r.t. time and solution quality

only convex MINLPs:

- SBB or DICOPT with IPOPT or KNITRO solves more than with CONOPT
- AlphaECP superior to Bonmin superior to SBB and DICOPT w.r.t. solution qual.
- AlphaECP superior to SBB superior to Bonmin and DICOPT w.r.t. time

Thank you!

PS: COIN-OR sessions TD-48 and TE-48 today 15:00-18:30 in room RB 213

210 MINLPs:

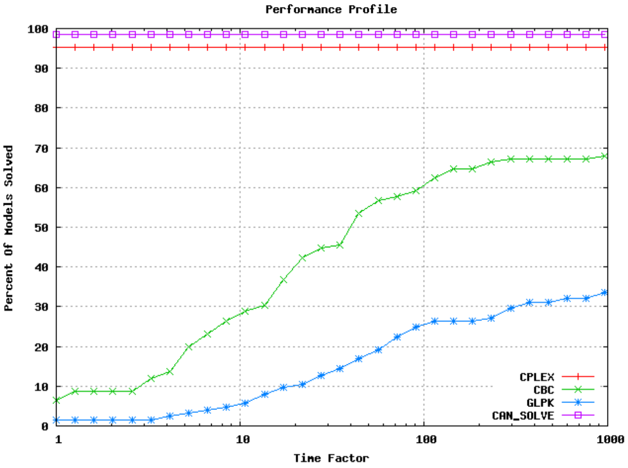
solver	feasible	infeasible	unbounded	fail
SBB+CONOPT	136	6	1	67
SBB+IPOPT	132	3		75
SBB+KNITRO	112	10		85
SBB+CONOPT+IPOPT	139	2	1	68
DICOPT+CONOPT	111	4		95
DICOPT+IPOPT	130	2		78
DICOPT+KNITRO	118	19		73
DICOPT+CBC	120	6		84
AlphaECP+CPLEX	149			61
AlphaECP+CBC				
Bonmin				

86 convex MINLPs:

solver	feasible	infeasible	unbounded	fail
SBB+CONOPT	38		1	47
SBB+IPOPT	42			44
SBB+KNITRO	38			48
SBB+CONOPT+IPOPT	42		1	43
DICOPT+CONOPT	41			45
DICOPT+IPOPT	45			41
DICOPT+KNITRO	42	4		40
AlphaECP+CPLEX	79			7
Bonmin				

MIPs from LINLib - Performance profile

all MIPs from LINLib ⇒ 125 models



solved = found best solution among all solvers