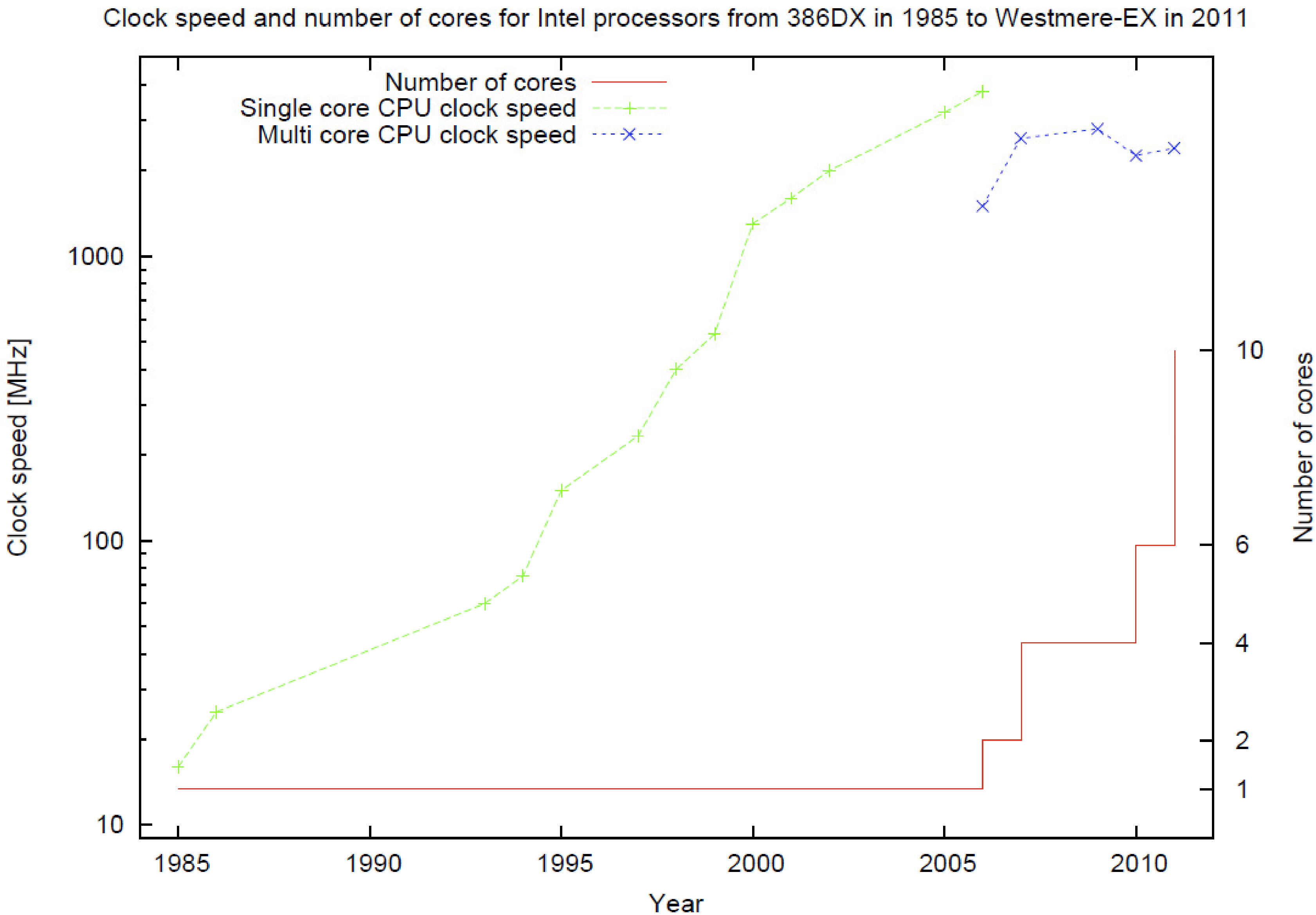
DFG Research Center MATHEON
Mathematics for key technologies

What Could a Million Cores Do To Solve Integer Programs?

A supercomputer is a machine for turning a compute-bound problem into an I/O-bound problem. --Ken Batcher

Thorsten Koch   Yuji Shinano
Zuse Institute Berlin / MATHEON
Ted Ralphs / Lehigh

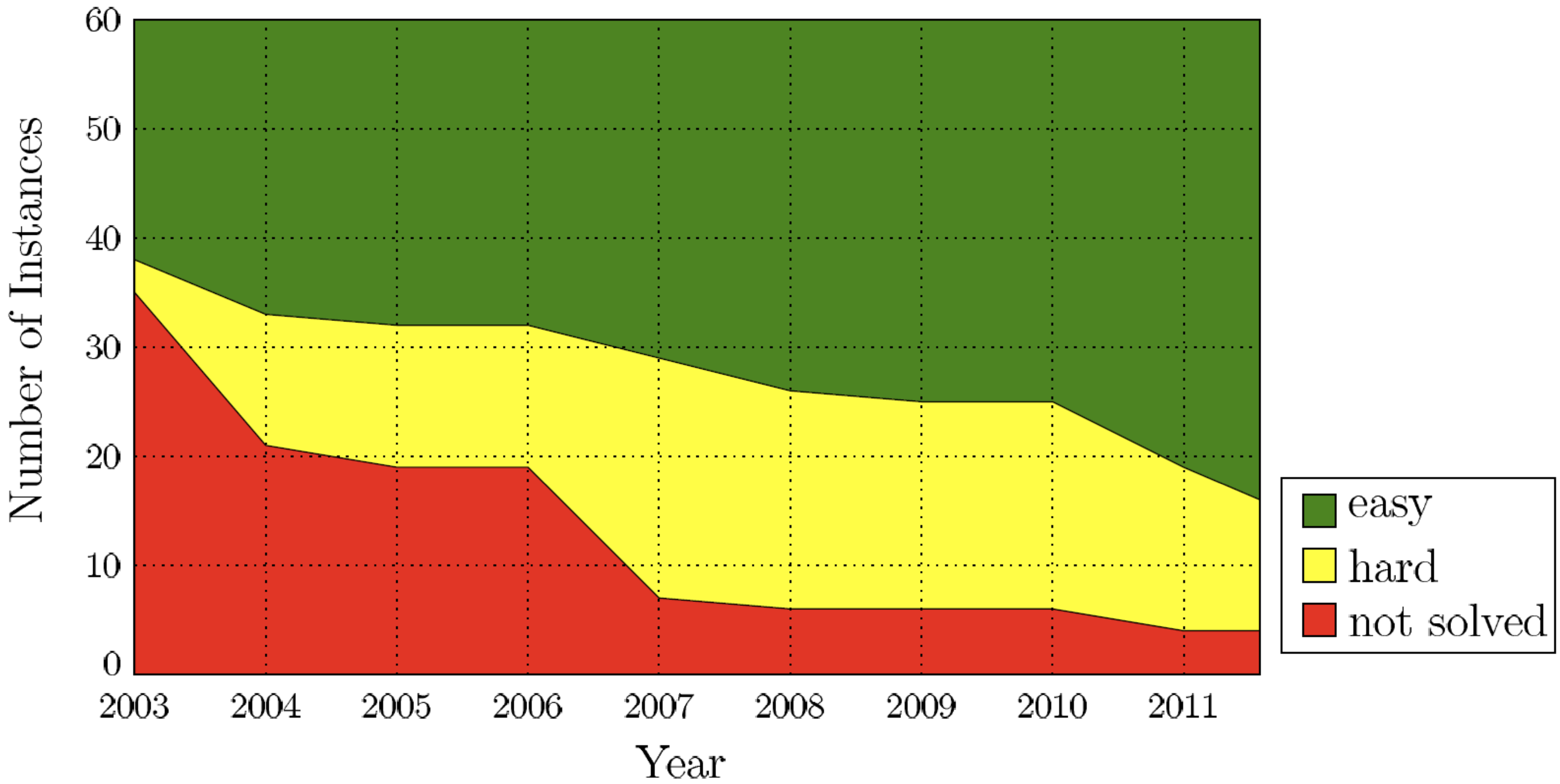
Clock speed / number of cores per CPU



What are the reasons why MIPs cannot be solved ?

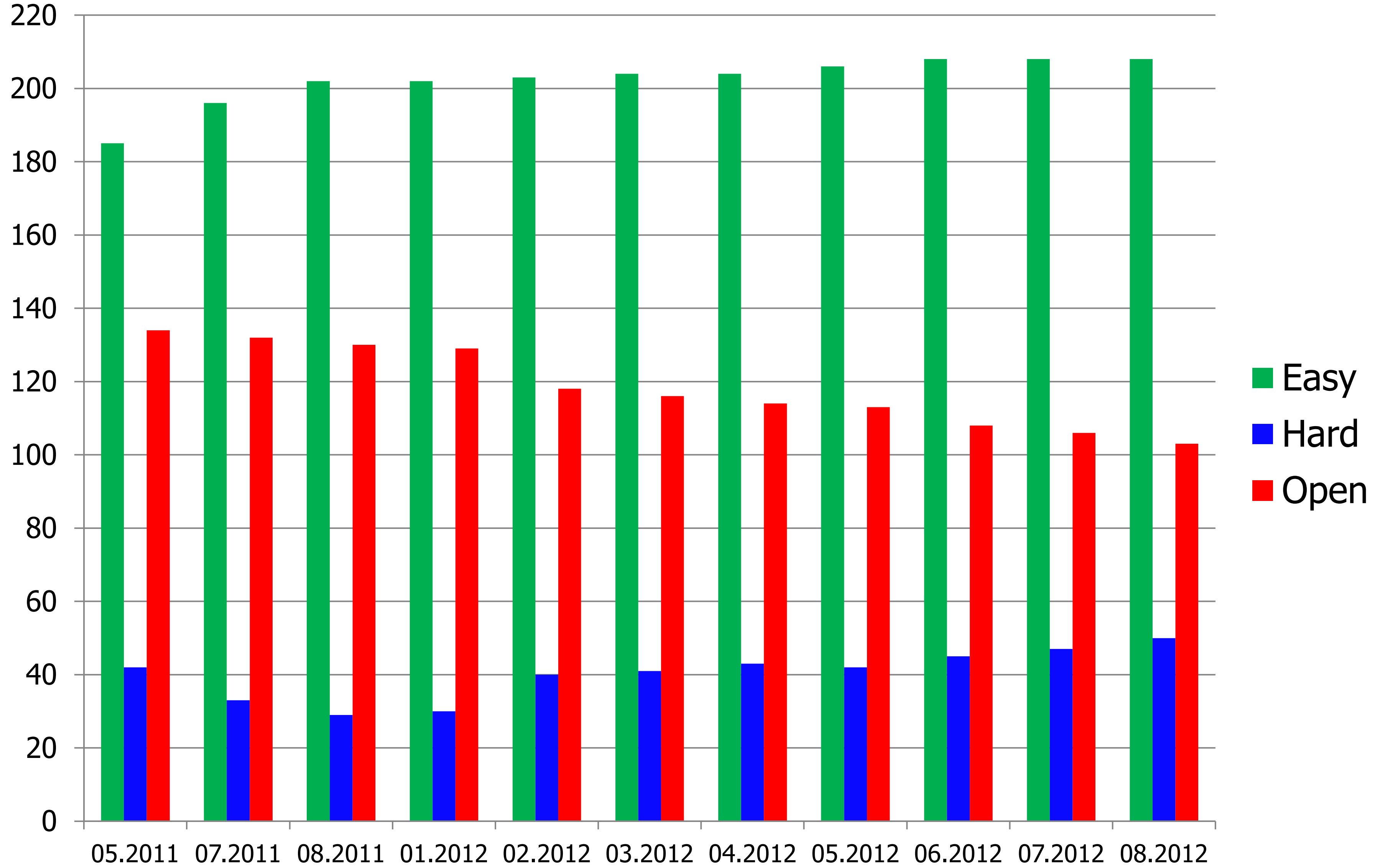
1. Genuine bad formulation
2. Bad dual bounds
3. LP is difficult/slow, especially reoptimizing
4. Bad numerical properties
5. Difficult to find primal solution
6. Large enumeration tree, e.g. due to symmetry
7. Just big
8. Nobody knows

Development of MIPLIB 2003

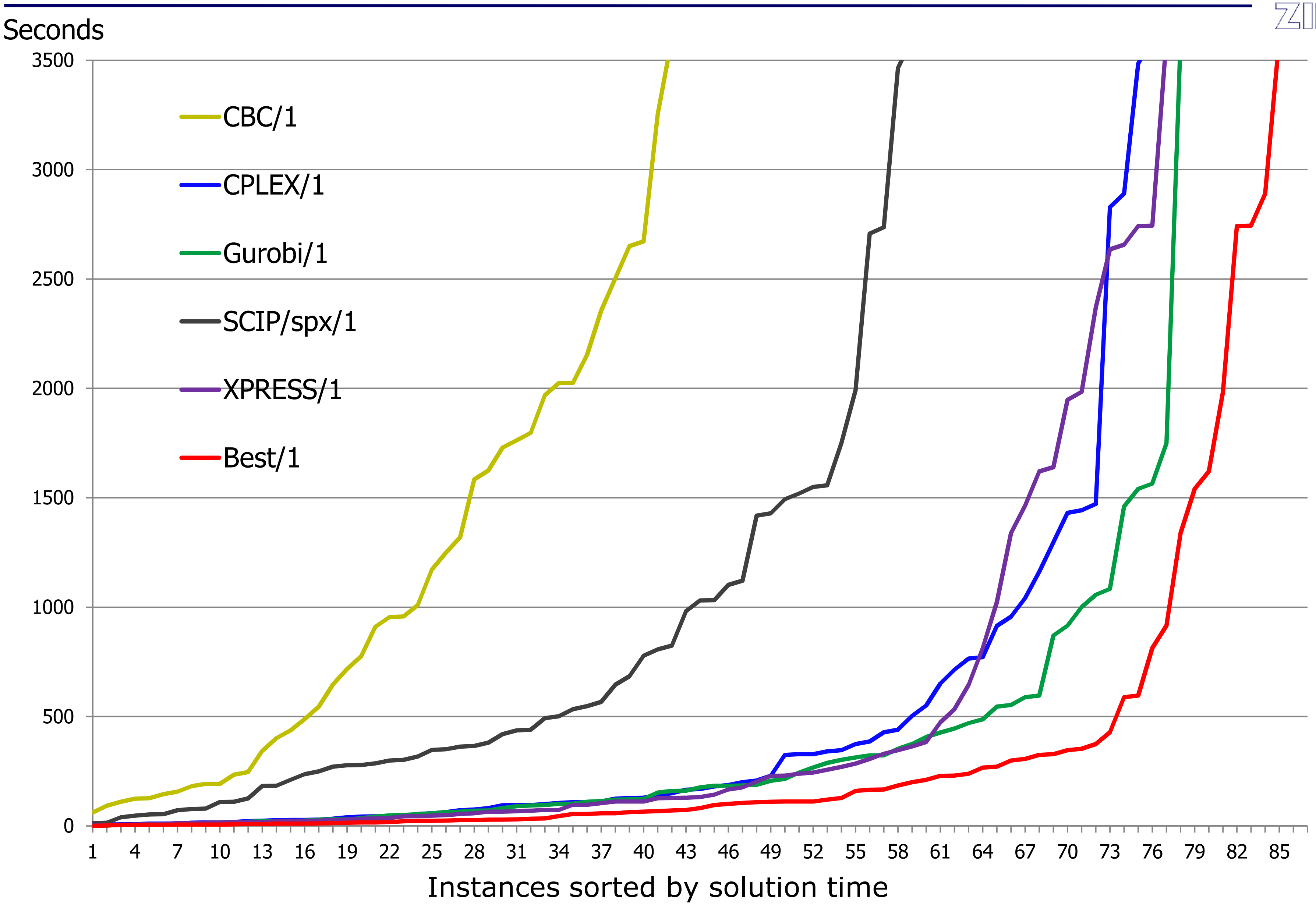


easy can be solved within an hour on a contemporary pc with a state-of-the-art solver
hard are solvable but take a longer time or require specialized algorithms
open, *i.e. not solved* instances for which the optimal solution is not known

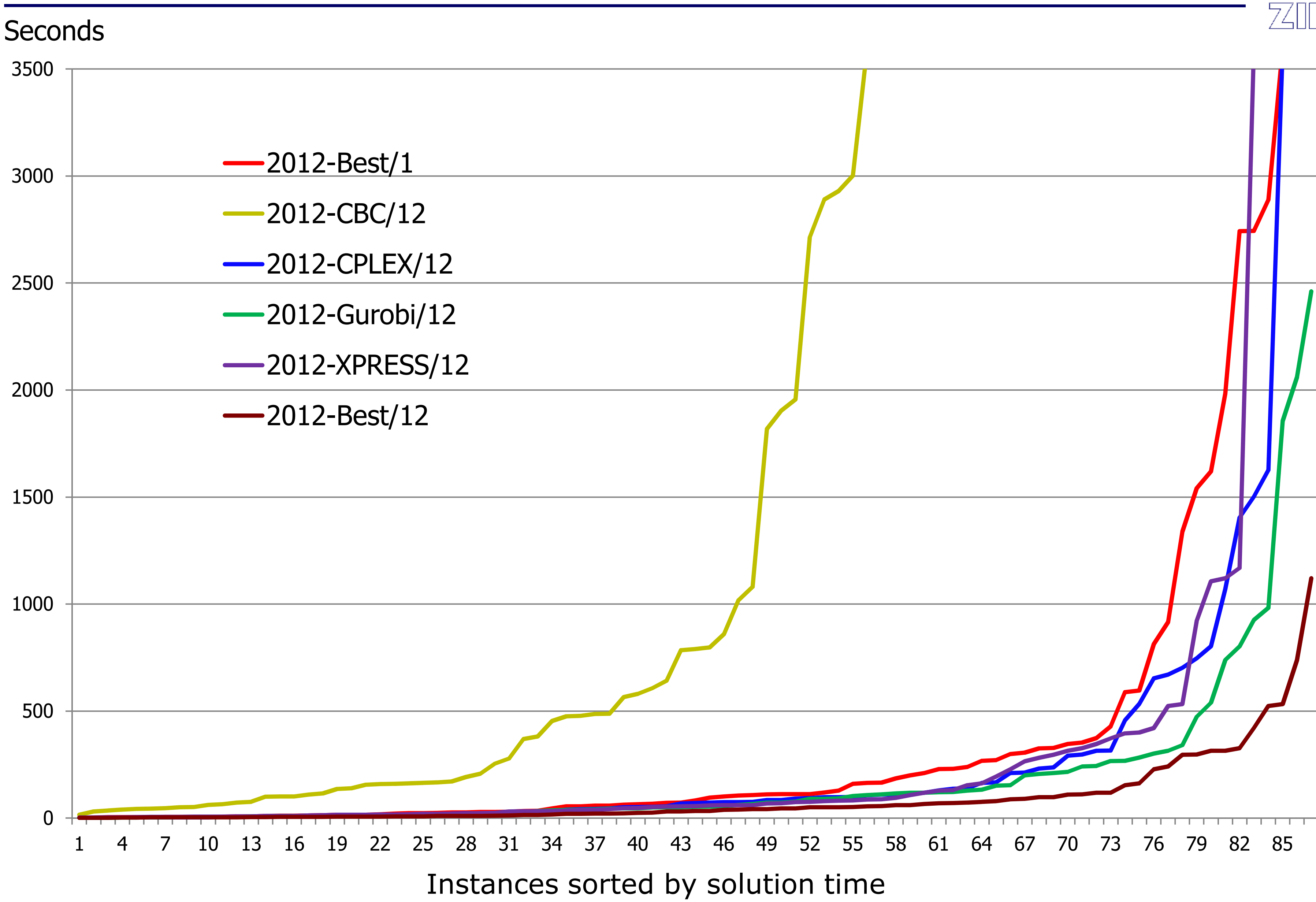
Development of MIPLIB 2010



2012 Performance on MIPLIB 2010 / 1 thread



2012 Performance on MIPLIB 2010 / 12 threads



What Does a Million Core System Look Like?

The K Computer

What does a million core system look like?

#	Computer	Year	Cores	Processor Family	MHz	Cores	OS
1	NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C	2010	186,368	EM64T	2930	6	Linux
2	Cray XT5-HE Opteron 6-core 2.6 GHz	2009	224,162	x86_64	2600	6	Linux
3	Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU	2010	120,640	EM64T	2660	6	Linux
4	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU	2010	73,278	EM64T	2930	6	Linux
5	Cray XE6 12-core 2.1 GHz	2010	153,408	x86_64	2100	12	Linux
6	Bull bullx super-node S6010/S6030	2010	138,368	EM64T	2260	8	Linux
7	BladeCenter QS22/LS21 Cluster, PowerXCell 8i / Opteron	2009	122,400	Power	3200	9	Linux
8	Cray XT5-HE Opteron 6-core 2.6 GHz	2009	98,928	x86_64	2600	6	Linux
9	Blue Gene/P Solution	2009	294,912	Power	850	4	CNK/SLES
10	Cray XE6 8-core 2.4 GHz	2010	107,152	x86_64	2400	8	Linux

Cluster composed of Processing Elements (PE), lets assume

4 CPUs with 32 cores = 256 cores per PE with shared memory.

Having 4000 PE = 1 Million cores.

=> Next generation Super Computer will have it!

1	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom / 2011 IBM	1572864	16324.75	20132.66	7890.0
2	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
3	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	786432	8162.38	10066.33	3945.0
4	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR / 2012 IBM	147456	2897.00	3185.05	3422.7
5	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
6	DOE/SC/Oak Ridge National Laboratory United States	**Jaguar** - Cray XK6, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA 2090 / 2009 Cray Inc.	298592	1941.00	2627.61	5142.0
7	CINECA Italy	**Fermi** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	163840	1725.49	2097.15	821.9
8	Forschungszentrum Juelich (FZJ) Germany	**JuQUEEN** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	131072	1380.39	1677.72	657.5
9	CEA/TGCC-GENCI France	**Curie thin nodes** - Bullx B510, Xeon E5-2680 8C 2.700GHz, Infiniband QDR / 2012 Bull	77184	1359.00	1667.17	2251.0
10	National Supercomputing Centre ...	**Nebulae** - Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband ...	120640	1271.00	2984.30	2580.0

How about memory?

- The memory hierarchy will become increasingly complex
  - L1 Cache 1×
  - L2 Cache 4×
  - L3 Cache 16×
  - Local / Remote Memory 50 – 700×
  - Network >1,000×
  - Disk >3,000,000×
- Total memory will increase
- Memory per PE will increase
- Memory per core will decrease
- Reliability will become a big issue:
  - “DRAM Errors in the Wild: A Large-Scale Field Study” Schroeder, Pinheiro, Weber, 2009
  - “Uncorrectable errors on 0.22% per DIMM per year makes a crash-tolerant application layer indispensable for large-scale server farms.”

Challenges

Measuring Performance

- Traditional measures are not appropriate.
  - The interesting problems are the ones that take too long to solve sequentially.
  - Need to account for the possibility of failure.
- It’s exceedingly difficult to construct a test set
  - Scalability varies substantially by instance.
  - Hard to know what test problems are appropriate.
  - A fixed test set will almost surely fail to measure what you want.
- Results are highly dependent on architecture
  - Difficult to make comparisons
  - Difficult to tune parameters
- Hard to get enough time on large-scale platforms for tuning and testing.
- Results are non-deterministic!
  - Determinism can be a false sense of security.
  - Lack of determinism requires more extensive testing.

Performance variability

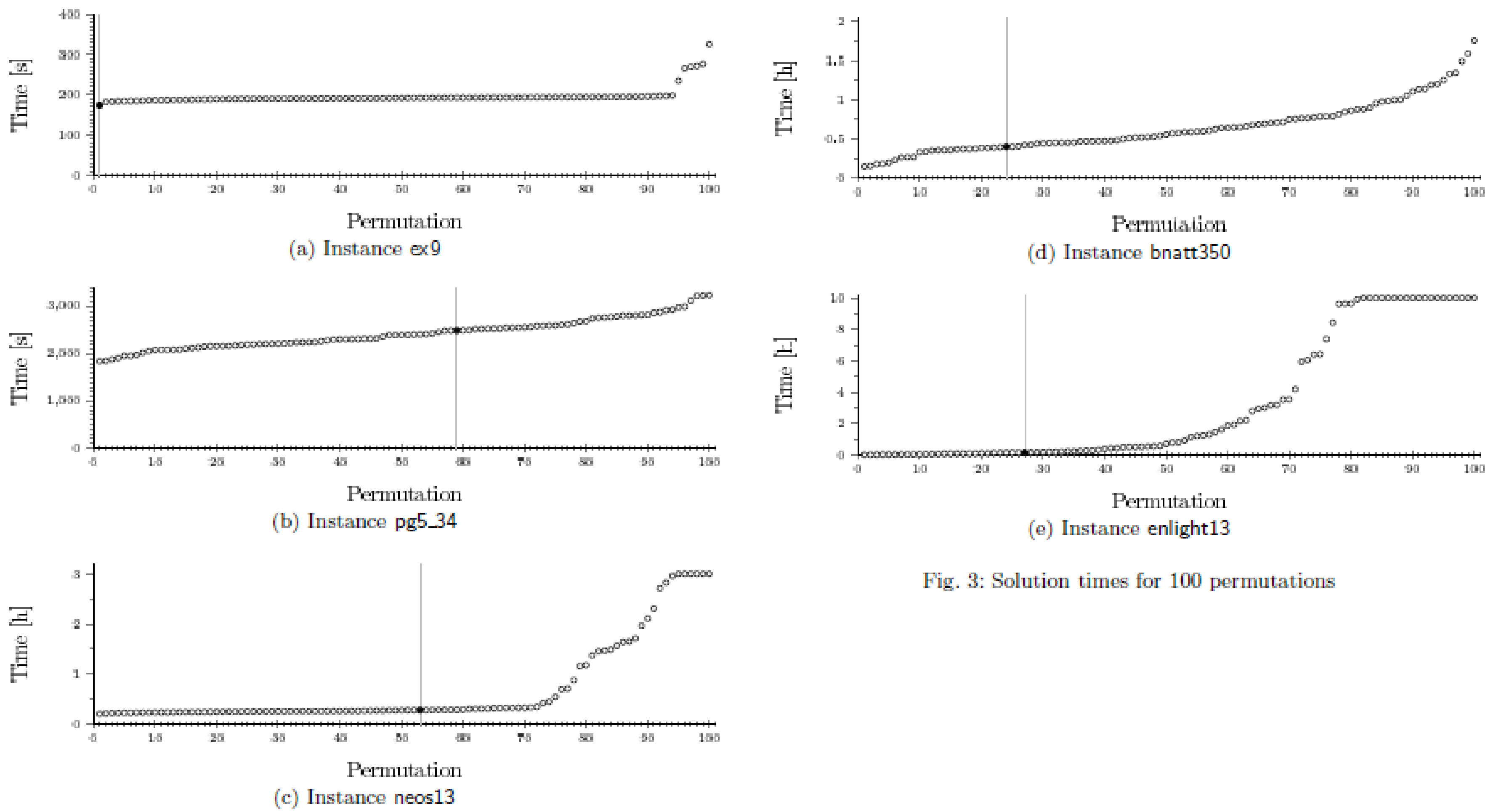
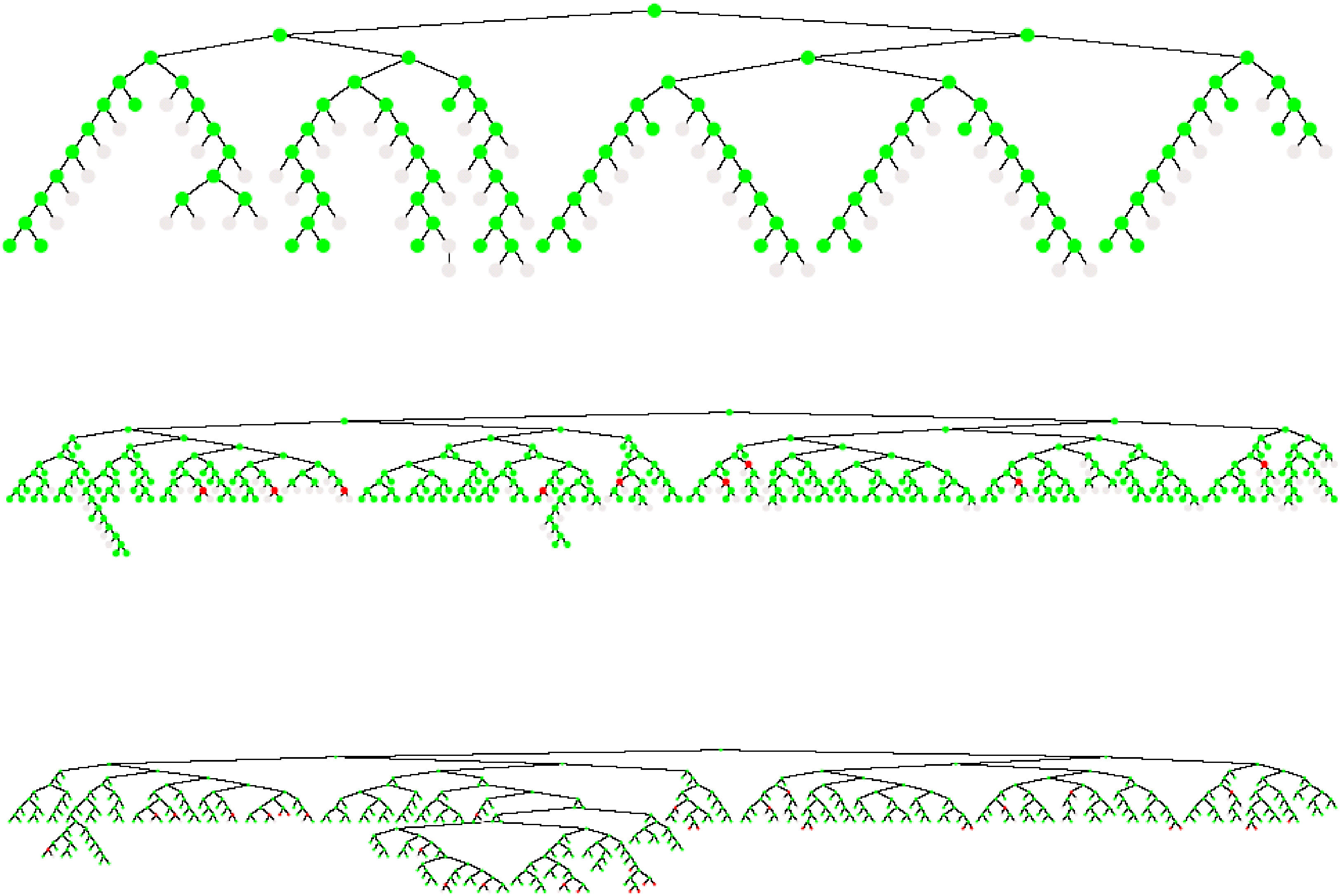
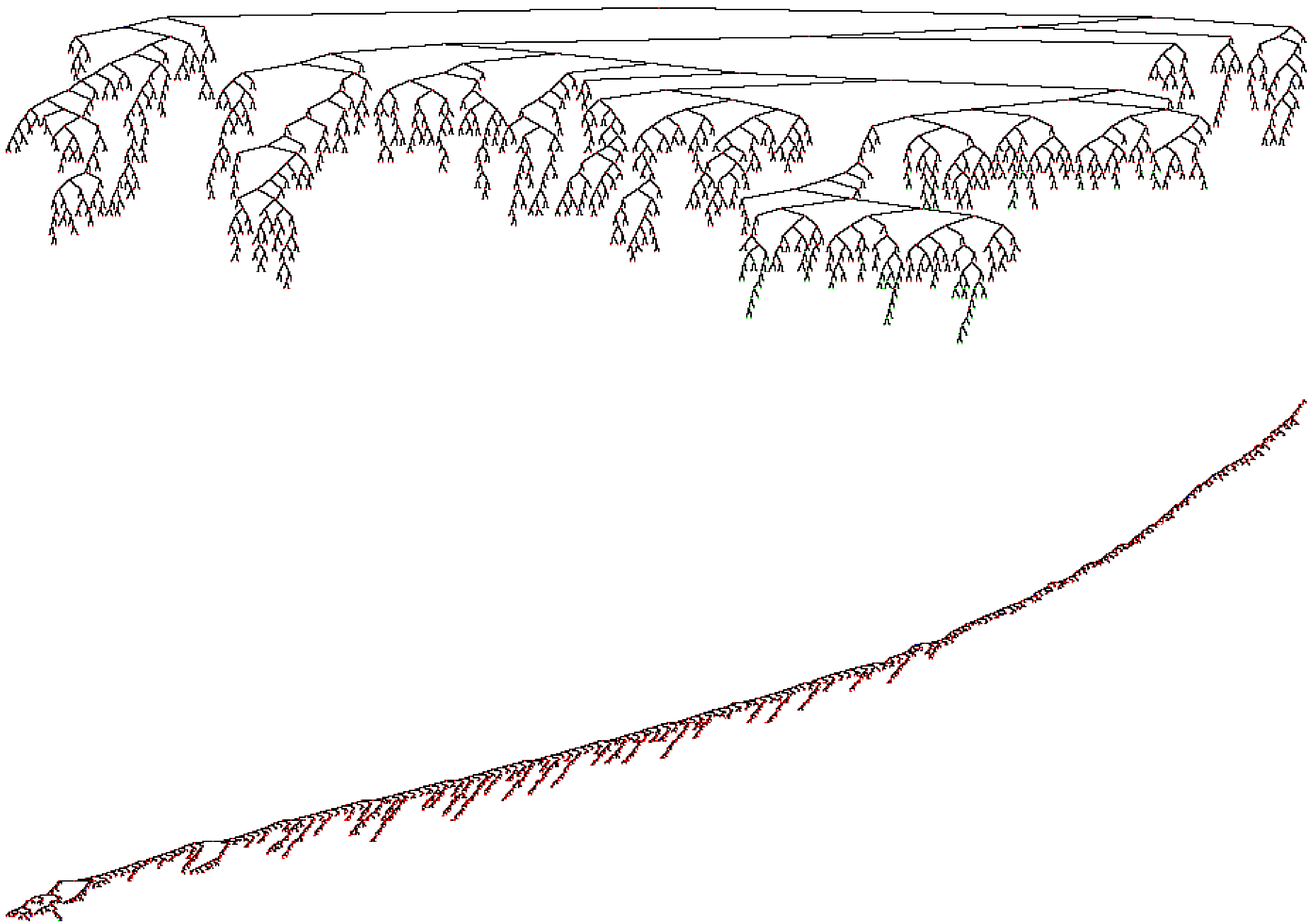


Fig. 3: Solution times for 100 permutations

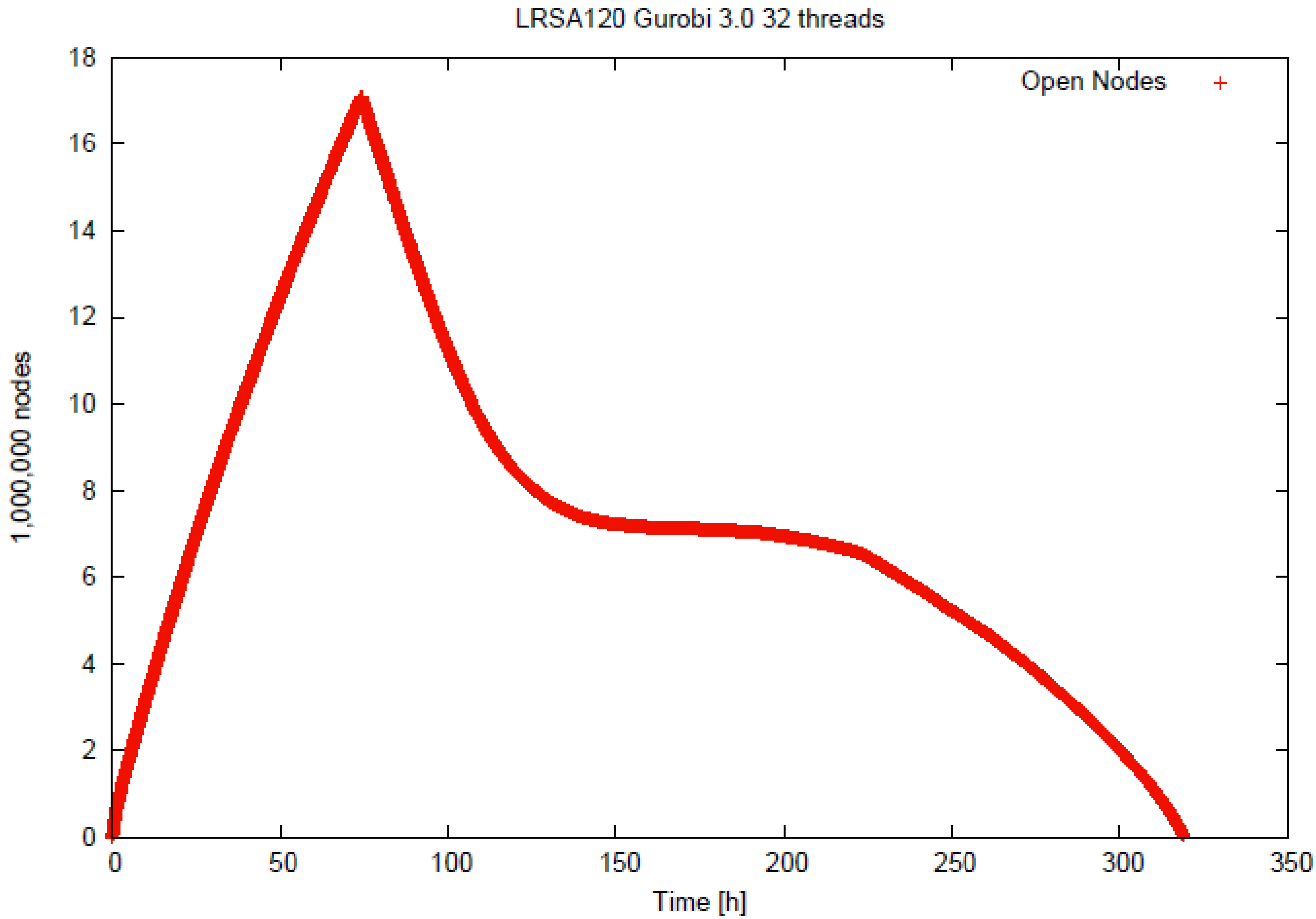
Ramp-up: Branching Trees



Ramp-up: Branching Trees



Typical? development of open nodes



Sources of Parallelism

Parallelization of tree search is easy in principle...but not in practice!

- Tree parallelism: Process different part of the tree simultaneously.
- Node Parallelization
  - Decomposition: Apply a decomposition approach in order to parallelize the bounding procedure.
  - Direct Parallelization: Parallelize solution of the LP relaxation
- Task parallelism
  - Special procedures for the ramp-up/ramp-down phases
  - Parallelize primal heuristics
  - Parallelize cut generation
  - Process multiple trees simultaneously

How to use the machine

- Each PE runs a standard shared memory MIP-Solver
- A multi-level master/slave load coordinator can be used to build and maintain a tree and distribute the problem over the 4000 PEs.

Problems:

- Ramp-up → racing ramp-up
- Ramp-down → phase detection
- PE runs out of memory: Node files, Iterated DFS
- Total number of open nodes limited
- How to check-point
- Parallel performance with increasing number of cores

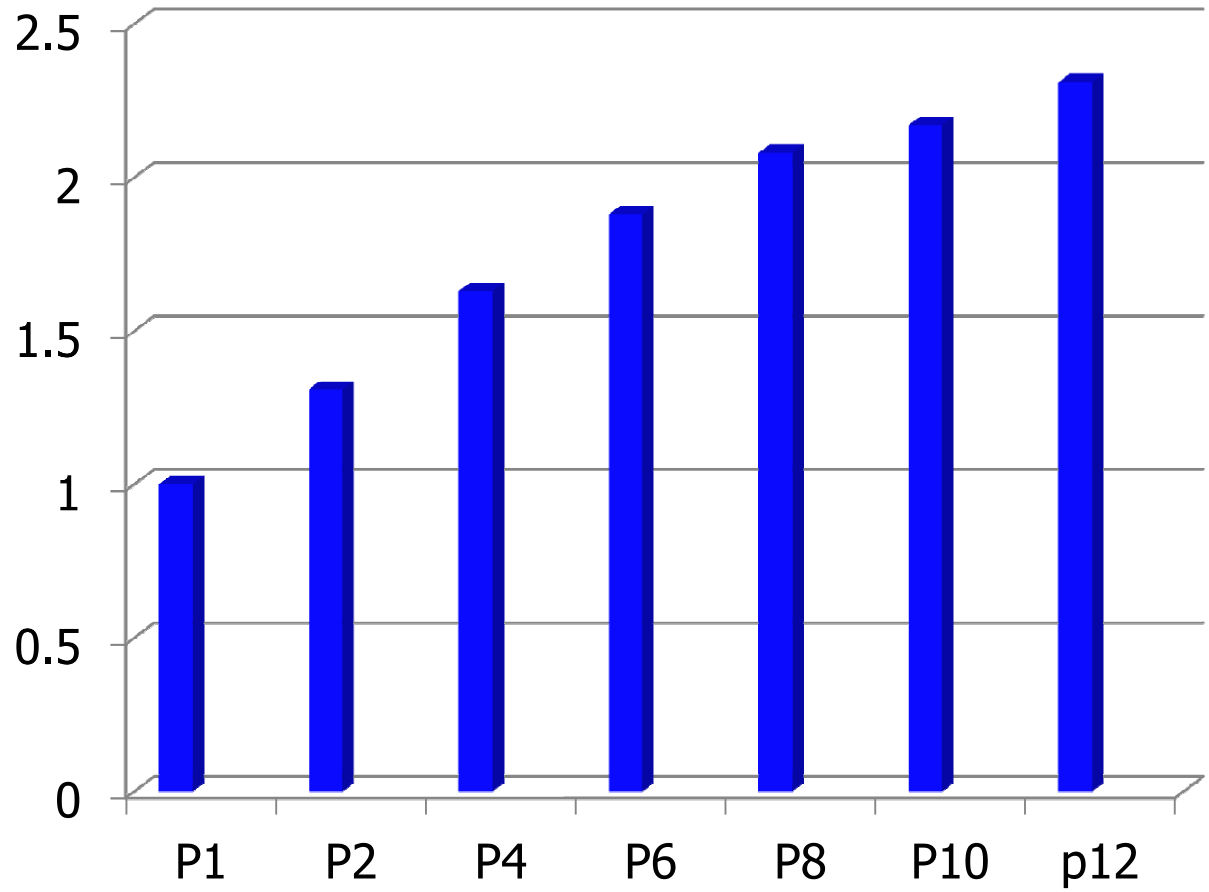
Solution on a Single PE

Shared Memory

Parallel Speedup over threads, Gurobi 3.0

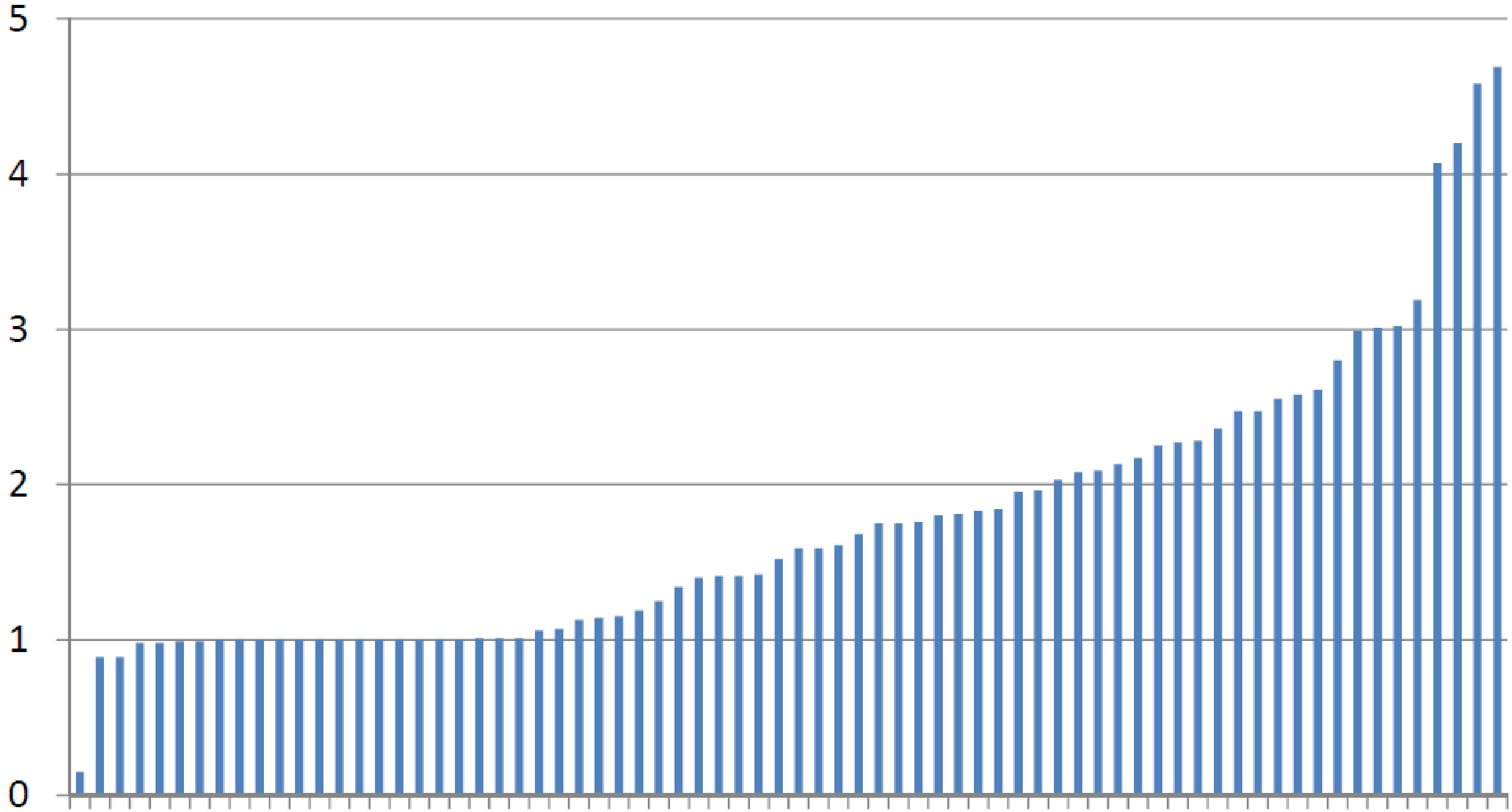
Slide courtesy of Ed Rothberg

#Threads	Speedup
1	1.0
2	1.31
4	1.63
6	1.88
8	2.08
10	2.17
12	2.31



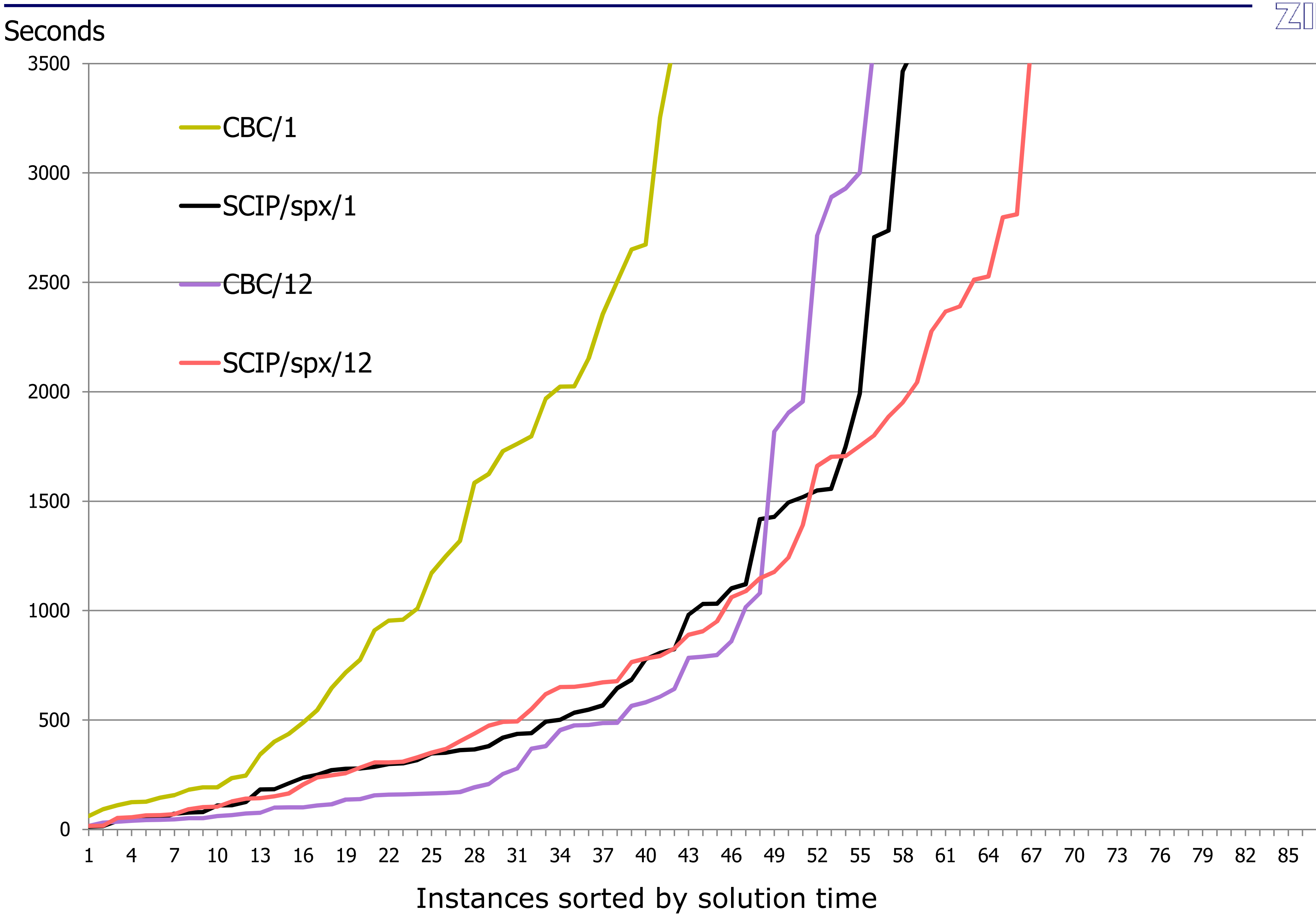
Speedups for P=4 using Gurobi (by Model)

Slide courtesy of Ed Rothberg

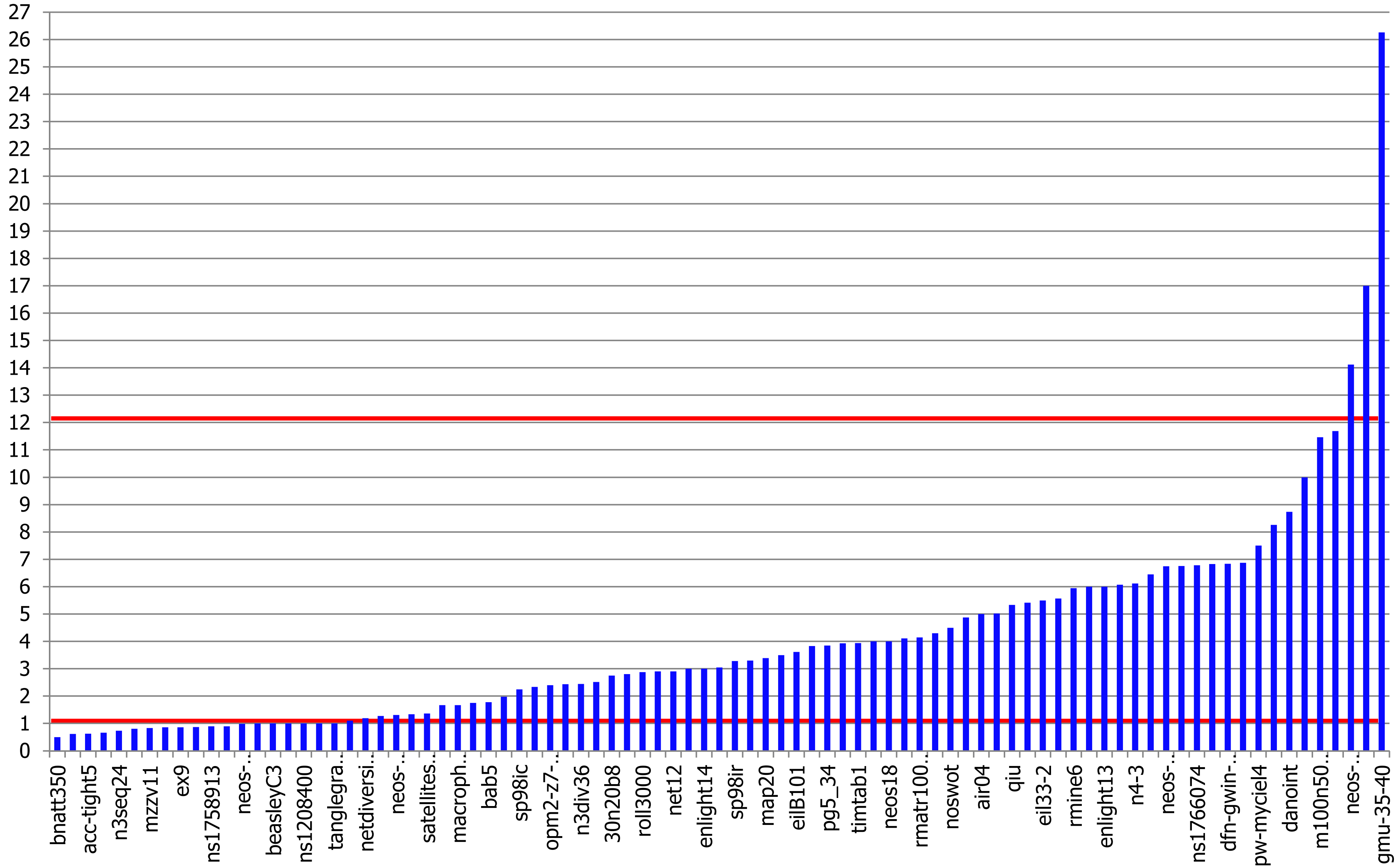


Average speedup from 1 to 12 threads on MIPLIB 2010 : 3 x

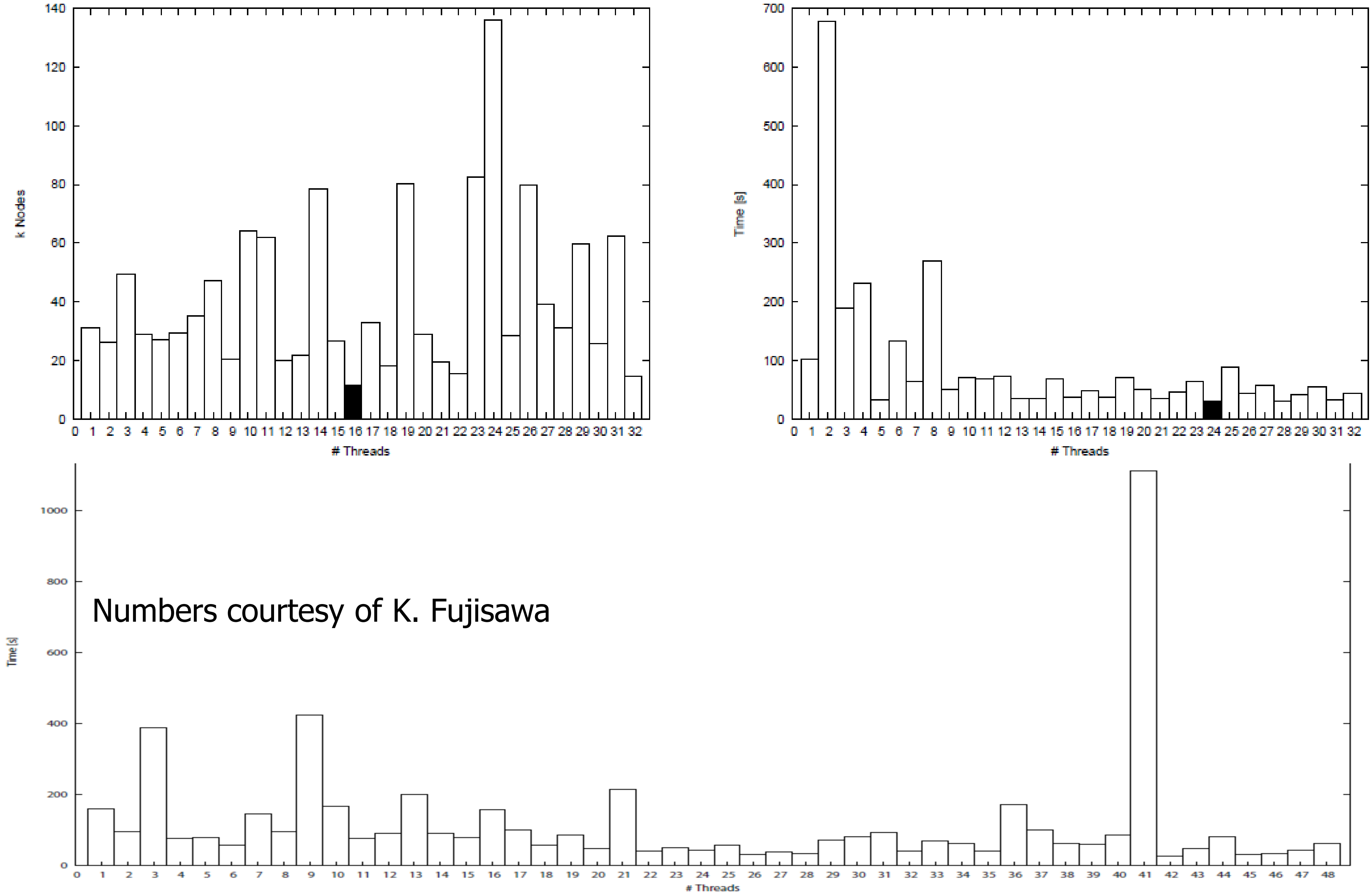
2012 Speedup 1:12 threads CBC/SCIP



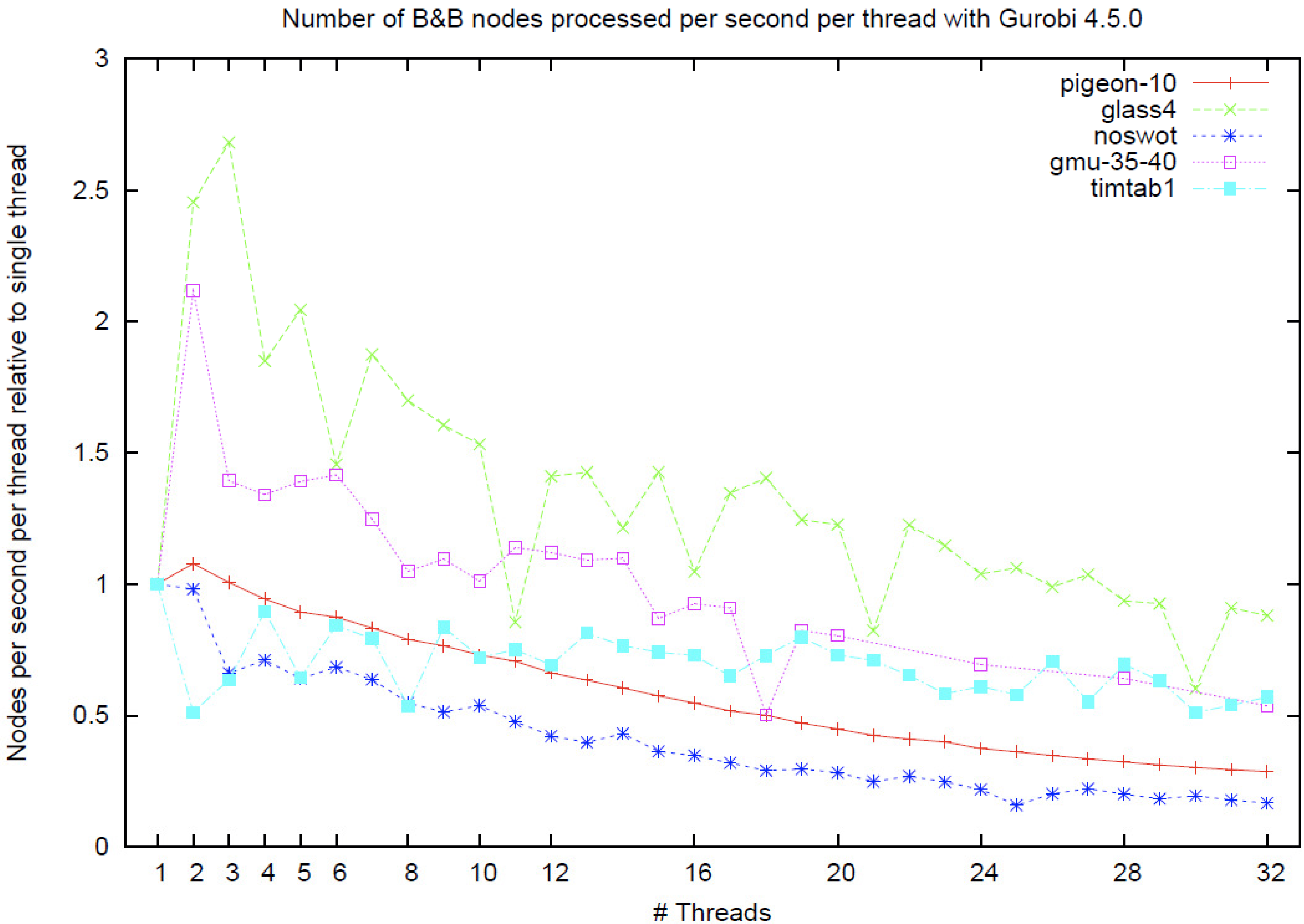
2012 Parallel speedup Best 1/12 per instance



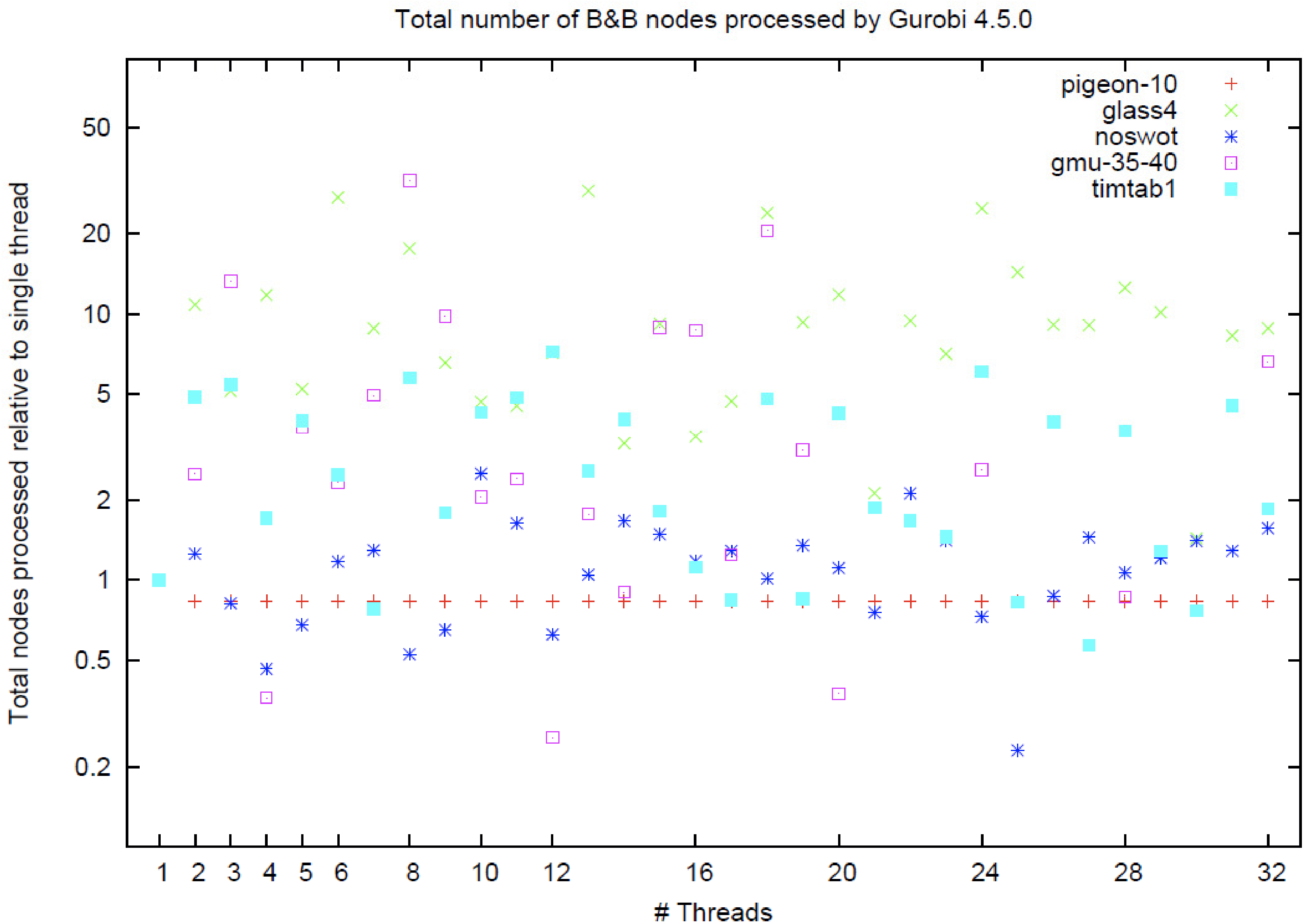
Performance relative to number of threads



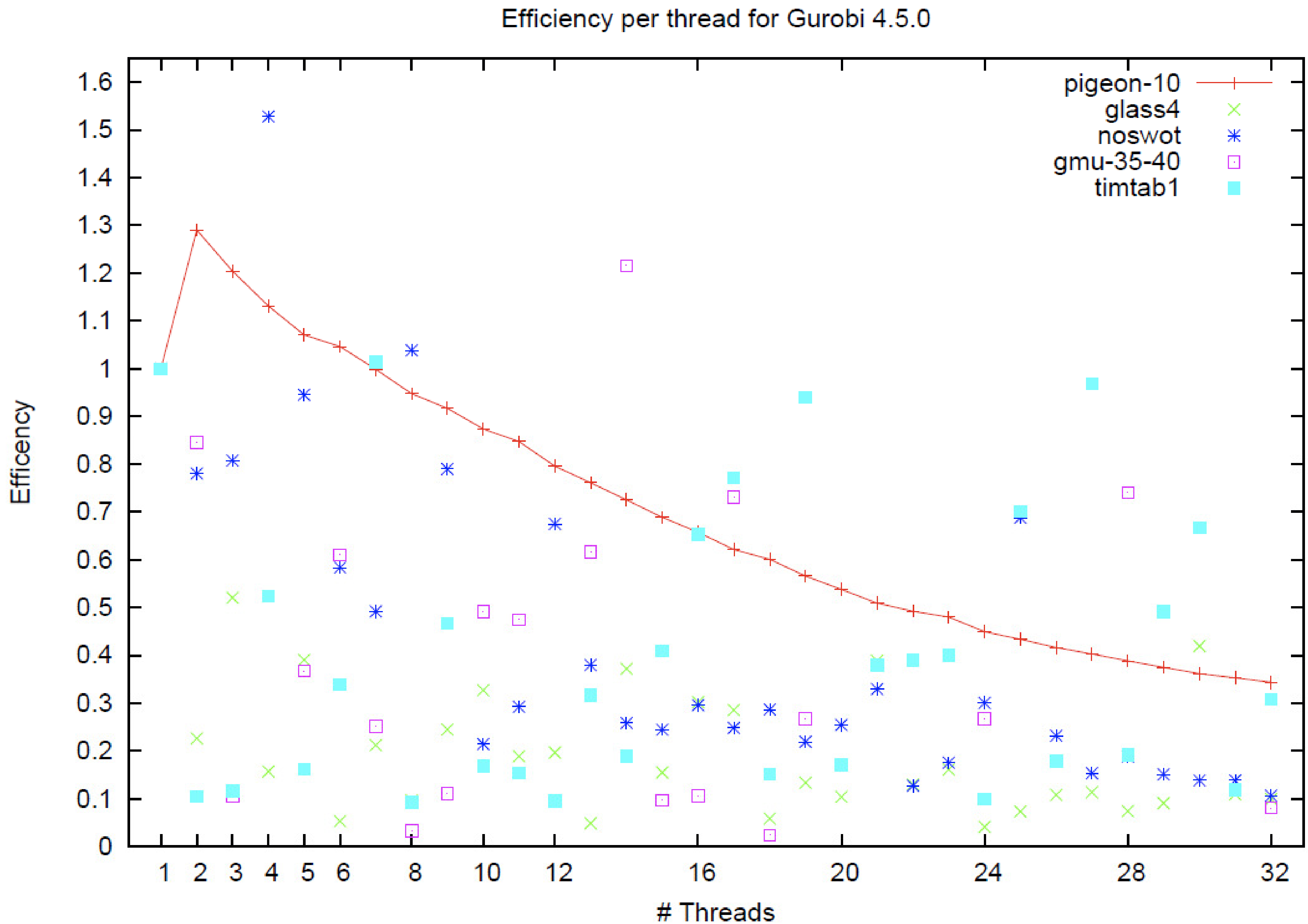
Nodes per second



B&B nodes


Total number of B&B nodes processed by Gurobi 4.5.0
pigeon-10
glass4
noswot
gmu-35-40
timtab1
Total nodes processed relative to single thread
50
20
10
5
2
1
0.5
0.2
1
2
3
4
5
6
8
10
12
16
20
24
28
32
Threads

Efficiency of MIP solvers



Notes

- Current shared memory solvers – in spite of considerable efforts – do not scale well! Especially, because of the root-phase
- Ramp-up might become a problem
- Better algorithms lead to fewer nodes
- Memory bandwidth unclear
- Running more than one sub-solver per PE is likely problematic because of memory consumption

Solution on Multiple PEs

Distributed Memory

What are the reasons why MIPs cannot be solved ?

1. Genuine bad formulation
2. Bad dual bounds
3. LP is difficult/slow, especially reoptimizing
4. Bad numerical properties
5. Difficult to find primal solution
6. Large enumeration tree, e.g. due to symmetry
7. **Just big**
8. Nobody knows

Assume we want to solve one LP per PE

Bottleneck LP Solver

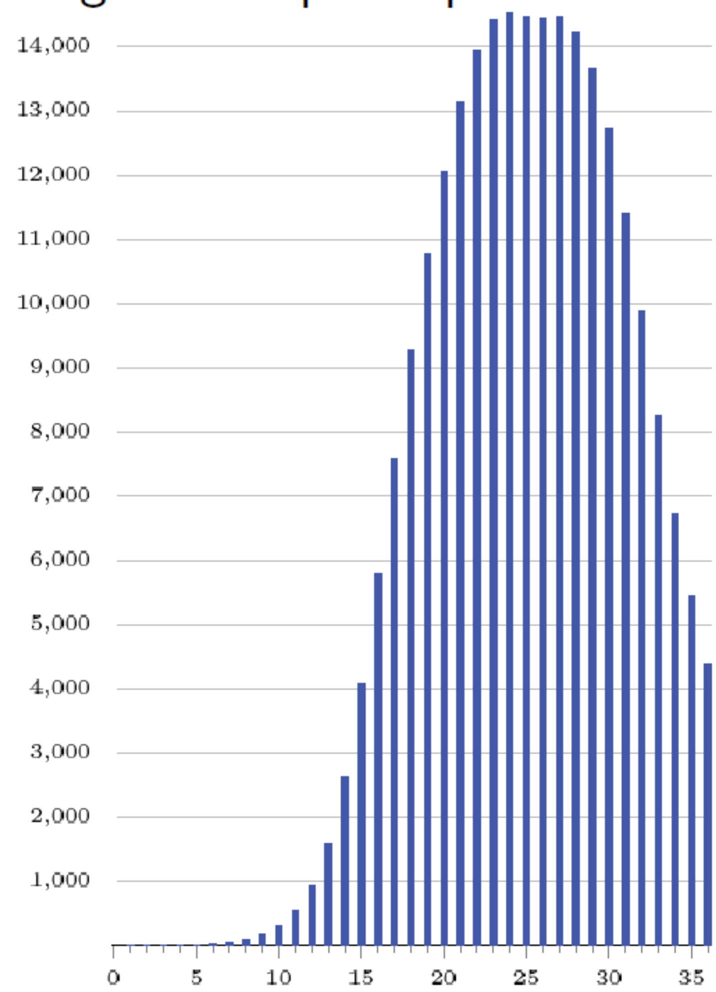
	Simplex	Barrier	Lagrange
Warmstart speedup	500 x	3 x	2 x ?
Parallel speedup	1 x	16+ x	8 x ?
Speed	1	0.6	0.1 ?
Needs crossover	no	yes	yes
Memory	1 x	up to 10 x	0.8 ?
Implemented	yes	no warmstart	no crossover
Numerically stable	most	good	unclear BIP likely

Crossover is single thread simplex and will likely take as long as simplex reoptimization (e.g. 20% of total time on 12 threads).
The Lagrange/Bundle/Volume algorithms only compute lower bounds on the solution.

Simplex reoptimization

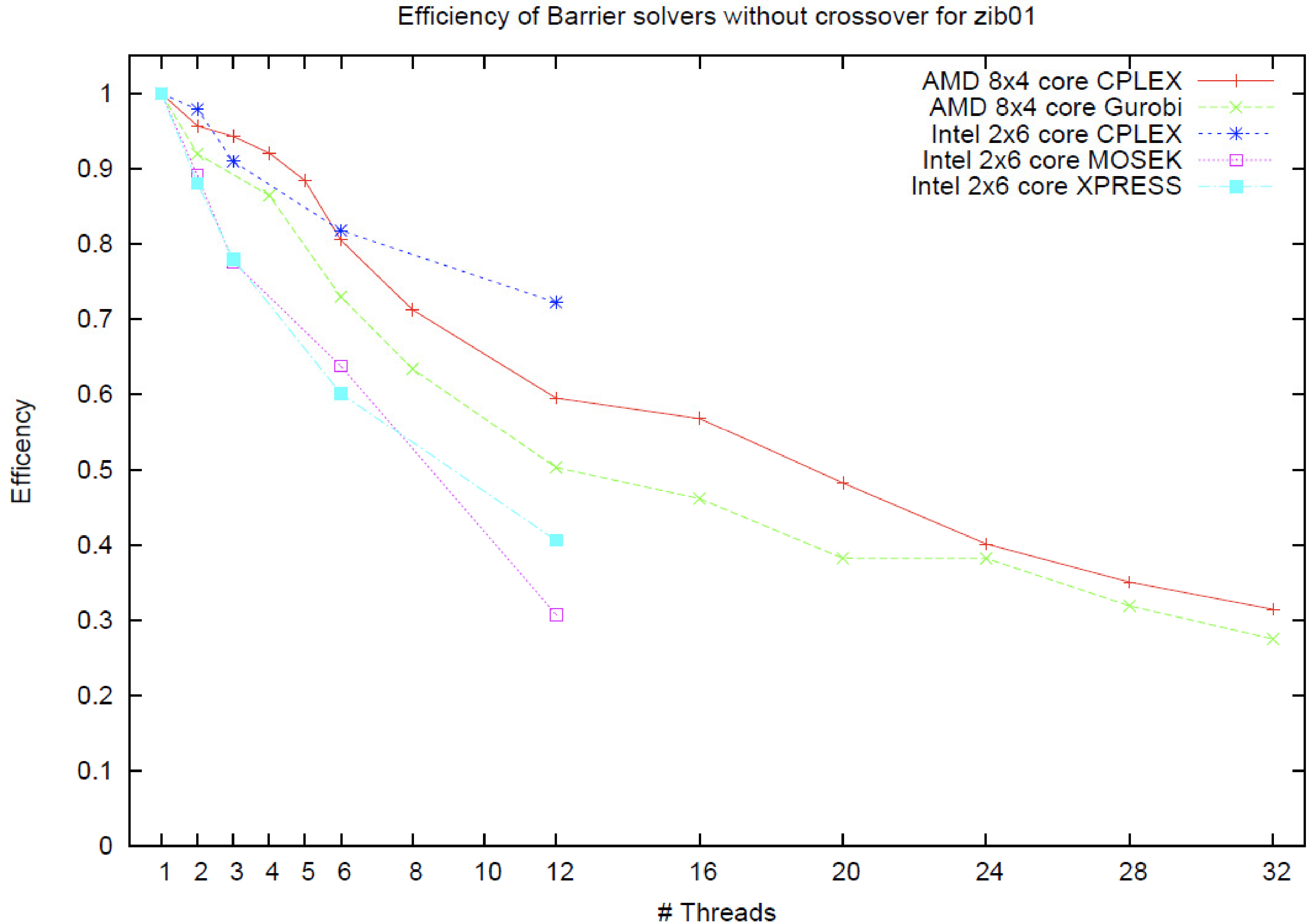
Depth	Iter	Depth	Iter
0	**3373.3**	16	10.4
1	34.1	17	9.4
2	29.9	18	8.7
3	26.4	19	8.3
4	24.2	20	7.9
5	22.0	21	7.4
6	20.5	22	7.3
7	19.6	23	7.3
8	18.6	24	6.8
9	17.9	25	6.6
10	16.9	26	6.3
11	16.5	27	6.1
12	15.8	28	5.8
13	14.7	29	5.5
14	13.7	30	5.1
15	13.3	31	4.6

Avg. nodes per depth



118 runs

Efficiency of Barrier solvers



Can we solve zib03 with a million cores?

We were not able to compute a feasible basis for zib03 so far.

After 10 h we still do not even have a primal feasible solution. Furthermore, experiments with smaller instances suggest the model is very unfavorable for the simplex method, especially regarding warm starts. Unfortunately, it is an IP.

Algorithm	Time [h]	Result	Memory [GB]	Resident [GB]
Primal Simplex	>300	Infeasibility 2189	24	18
Dual Simplex	>300	Lower bound 8335	24	18
Bundle	13	Lower bound 5951	55	18
Interior Point	103 (32 threads)	Optimal 1.2228.148	256	175
Crossover	>300	unfinished		

Some Predictions Pulled Out of the Air

- Expected Efficiencies
  - An efficiency of 0.2 seems achievable for an ILP solver on 128 core CPU
  - An efficiency of 0.1 seems realistic for solving ILPs on multiple PEs
  - An efficiency of 0.2 seems achievable for solving large LPs by barrier with crossover in parallel on 128 cores.
  - An efficiency of 0.1 seems possible for distributed solution of LPs by barrier.
- Expected Overall Performance
  - For a million core system with 128 cores per PE and 8000 Pes, this means a speed-up of 20000 in steady-state.
  - If we are only able to solve one LP relaxation (in parallel) on each PE due to size, then a speed-up of about 1000 is realistic.
  - If we are only able to solve the LP relaxations in distributed fashion using, say, 8 Pes, then we can only expect a speed-up of about 10.

Prediction is very difficult, especially about the future. --Niels Bohr

Summary

Symptom	HW cure	SW cure
Slow LPs	Faster cores	LP algorithm improvement
Many nodes	More cores	IP algorithm improvement
Big instances	More memory	

- Faster cores help in all cases (the way it used to be)
- More cores can improve slow LPs and big instances, too, but is limited due to memory bandwidth, sequential algorithms, and ramp-up/-down
- Better LP algorithms could help for big instances
- Better IP algorithms usually work against parallelization
- It seems unlikely that many instances that have weak bounds or poor formulations will be helped

Questions?

Conclusions

Slow LP, many nodes, big instances, ...

... chose 2 !

But what if the memory does not grow with the number of cores?

- Since a distributed Cholesky factorization is there
- a distributed Interior Point LP solver should be the solution. Only it will be even slower. Therefore,

We need more cores!

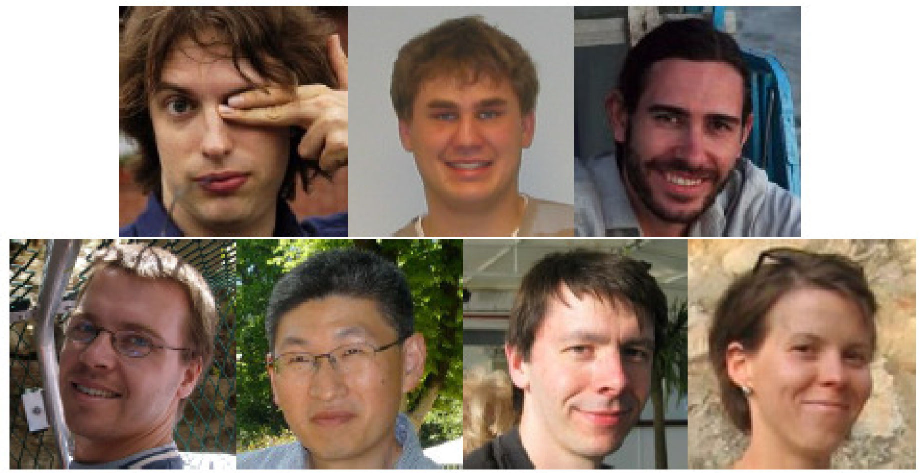
Thank you very much!

Questions?

ZIB IP Group

- ▷ Tobias Achterberg (IBM)
- ▷ Thorsten Koch
- ▷ Marc Pfetsch (TU Braunschweig)

- ▷ Timo Berthold
- ▷ Gerald Gamrath
- ▷ Ambros Gleixner
- ▷ Stefan Heinz
- ▷ Yuji Shinano
- ▷ Stefan Vigerske
- ▷ Kati Wolter



- ▷ Gregor Hendel
- ▷ Robert Waniek
- ▷ Michael Winkler

Mining the talks

Thanks for contributing slides to

Tobias Achterberg

Timo Berthold

Robert Bixby

Ed Rothberg

Zonghao Gu

Stefan Vigerske

and especially all I forgot

ZIB Instances

	Variables	Constraints	Non-zeros	Description
1	12,471,400	5,887,041	49,877,768	Group Channel Routing on a 3D Grid Graph (Chip-Bus-Routing)
2	37,709,944	9,049,868	146,280,582	Group Channel Routing on a 3D Grid Graph (different model, infeasible)
3	29,128,799	19,731,970	104,422,573	Steiner-Tree-Packing on a 3D Grid Graph
4	37,423	7,433,543	69,004,977	Integrated WLAN Transmitter Selection and Channel Assignment
5	9,253,265	9,808	349,424,637	Duty Scheduling with base constraints

Future algorithmic speed-up

- Most of the computing time in MIP solver is spend in the solution of the LP relaxations, in the parallel case especially in the root

- The Dual Simplex algorithm is the method of choice
- Little setup cost, fast on small/"easy" instances
- The Simplex algorithm has exponential complexity
- The Simplex algorithm is inherently difficult to parallelize

- Interior Point Methods have polynomial complexity
- In practice much less variation of solution times
- Interior Point Methods parallelize well
- BUT Interior Point Methods are not warm start capable

What is the MIPLIB

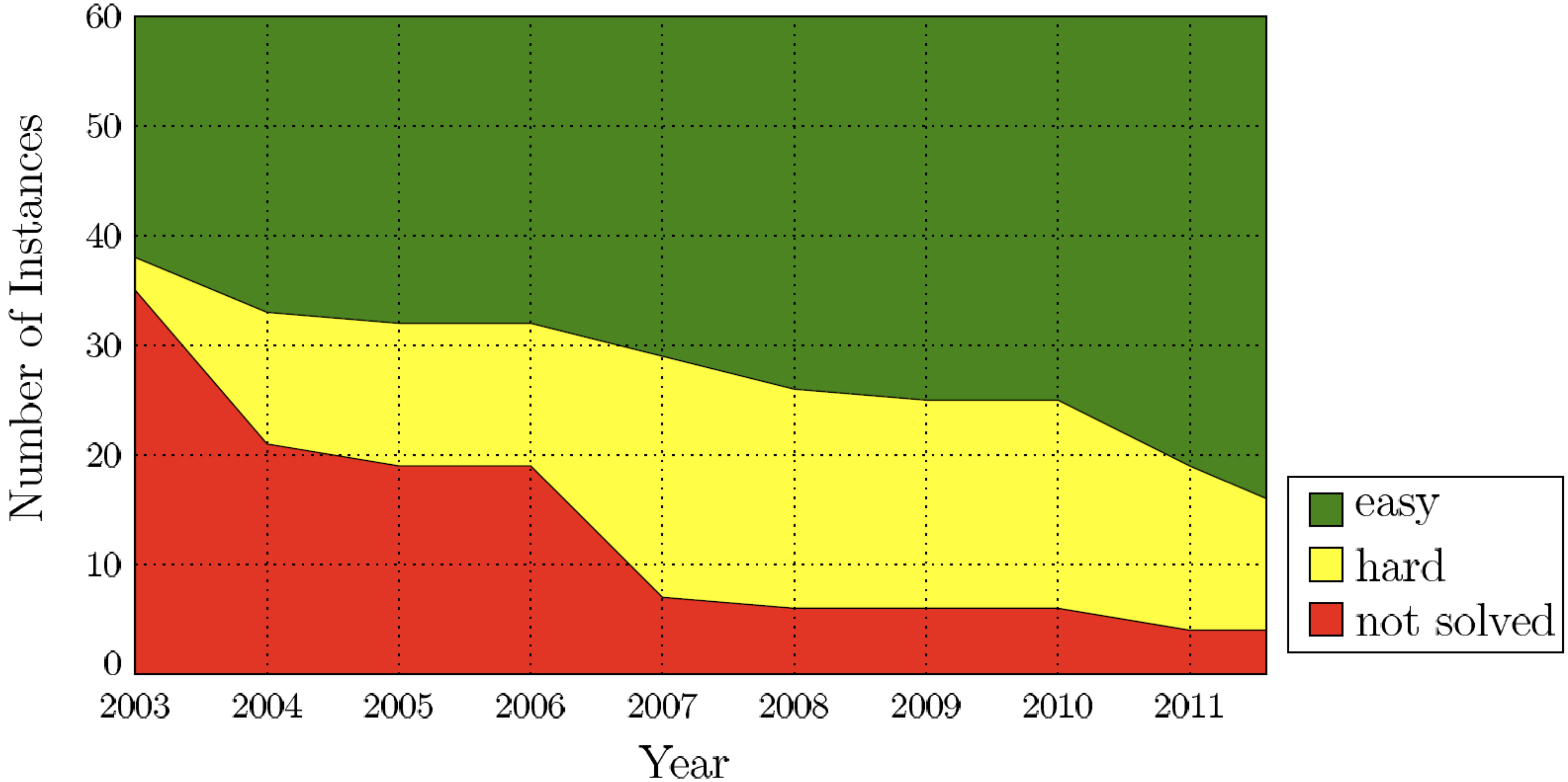
Version	Year	#	Who	Reference
1	1991	61	Bixby, Boyd, Indovina	
2	1992	61	Bixby, Boyd, Indovina	SIAM News 25, 15 (1992)
3	1996	65	Bixby, Ceria, McZeal, Savelsberg	Optima 58, 12-15 (1998)
4	2003	60	Achterberg, Koch, Martin	ORL 34(4) 361-372 (2006)
5	2010	87/361	Koch, Achterberg, Andersen, Bastert, Berthold, Bixby, Danna, Gamrath, Gleixner, Heinz, Lodi, Mittelmann, Ralphs, Salvagnin, Steffy, Wolter	MPC 3, 103-163 (2011)

The MIPLIB is a diverse collection of challenging real-world mixed integer programming (MIP) instances from various academic and industrial applications suited for benchmarking and testing of MIP solution algorithms.

The *Benchmark* set consist only of instances that could be solved in 2010 within two hours on a high-end personal computer by at least two MIP solvers.

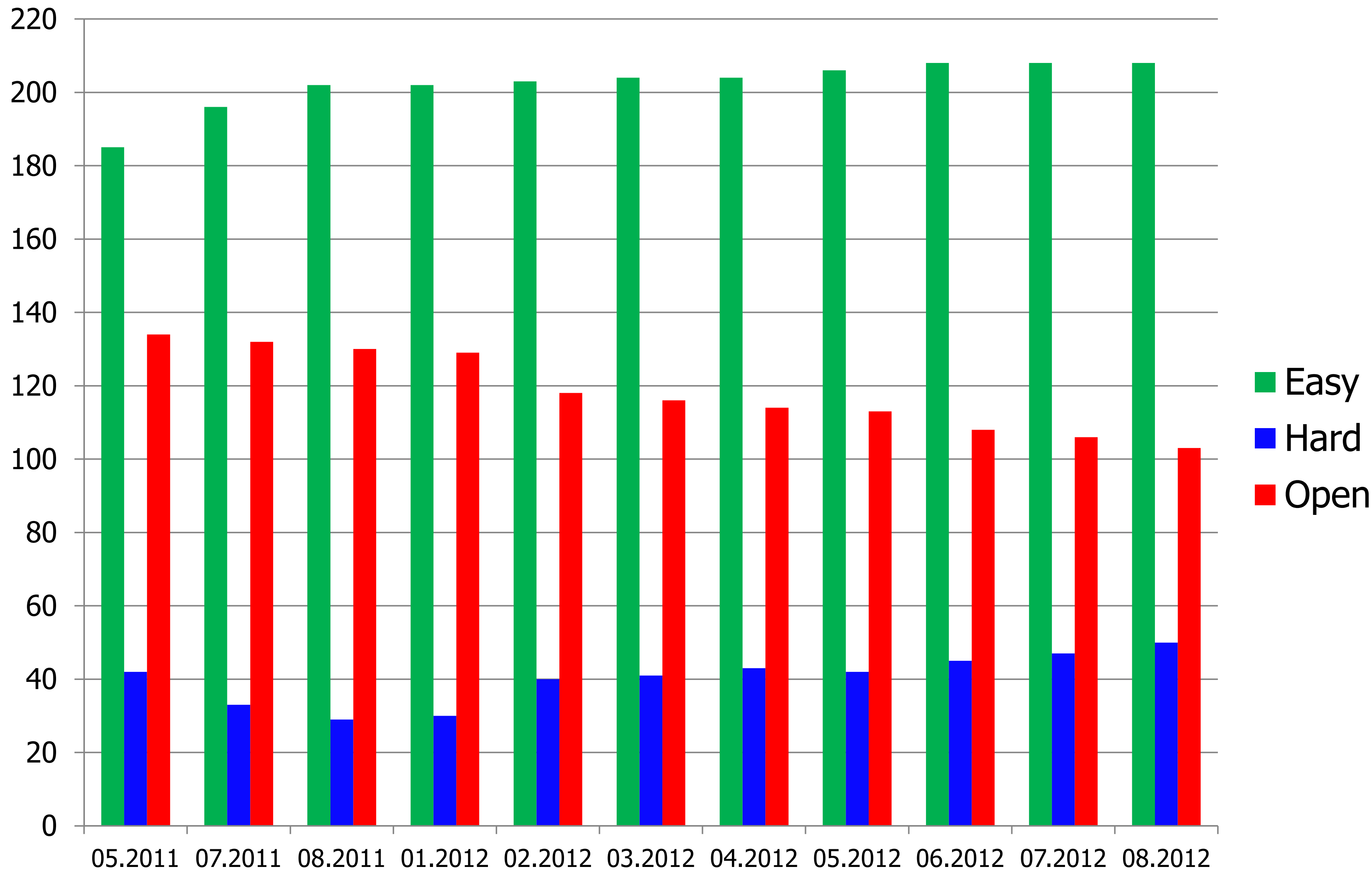
MIPLIB 5 still contains 2 instances from Version 2, 7 instances from Version 3, and 17 instances from Version 4.

Development of MIPLIB 2003



easy can be solved within an hour on a contemporary pc with a state-of-the-art solver
hard are solvable but take a longer time or require specialized algorithms
open, *i.e. not solved* instances for which the optimal solution is not known

Development of MIPLIB 2010

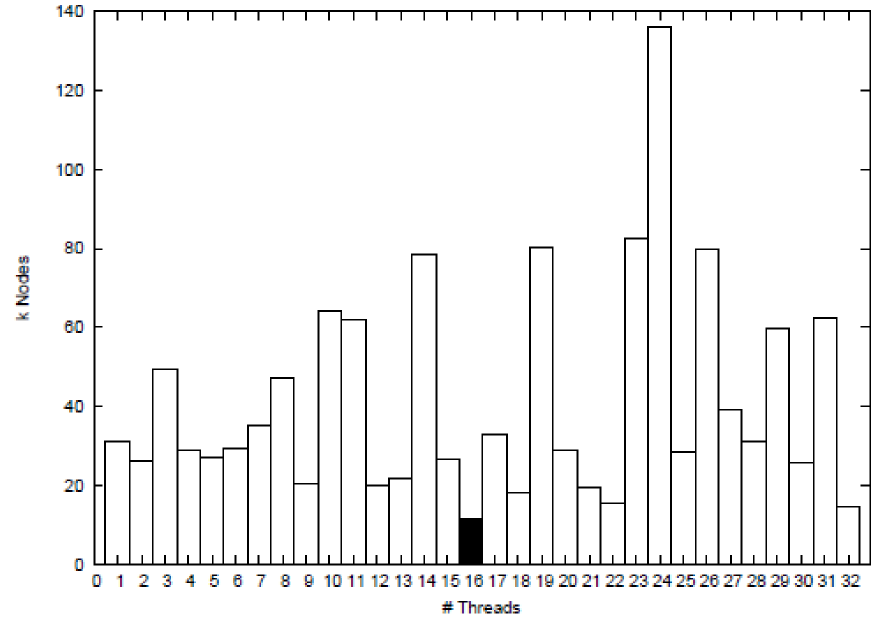


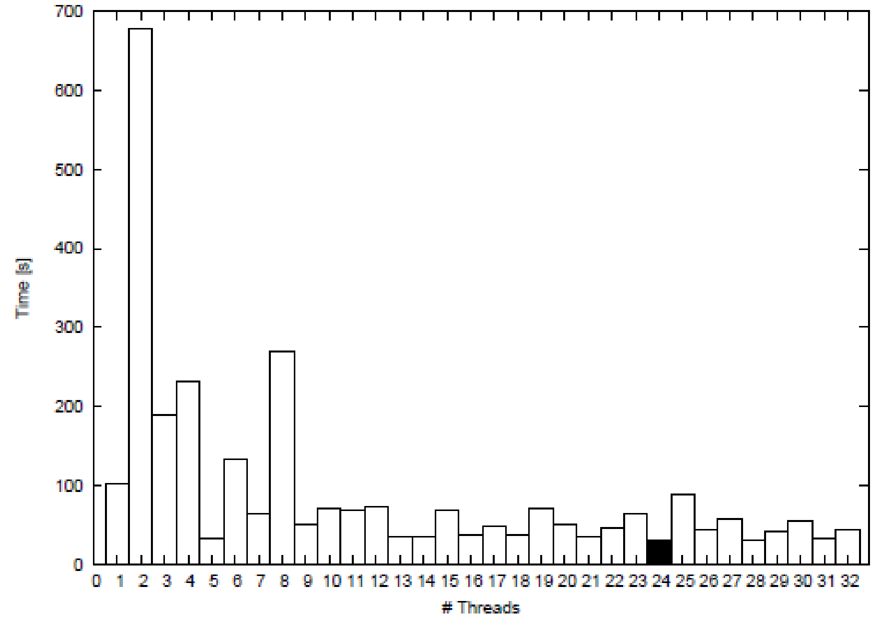
Some MIP Solvers then and now

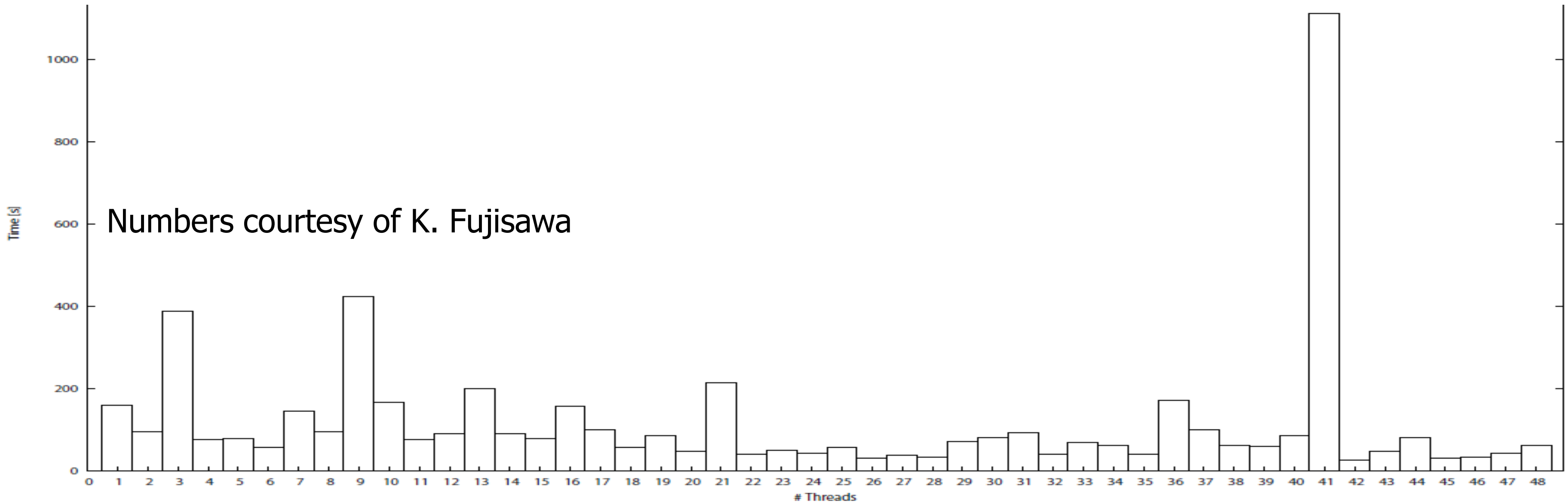
Solver	Version 2011	Version 2012	URL
IBM CPLEX	12.2	12.4	www.cplex.com
Gurobi	4.5 beta0	5.0	www.gurobi.com
FICO XPress-MP	7.2 RC	7.2	www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx
SCIP/spx	2.0.1	3.0.0	scip.zib.de
CBC	2.6.4	2.7.7	projects.coin-or.org/Cbc

Further solvers			
Glpk		4.47	www.gnu.org/software/glpk/glpk.html
lp_solve		5.5	lpsolve.sourceforge.net
Lindo		7.1	www.lindo.com
Minto		3.1	coral.ie.lehigh.edu/~minto
Symphony		5.4	projects.coin-or.org/SYMPHONY

Performance relative to number of threads

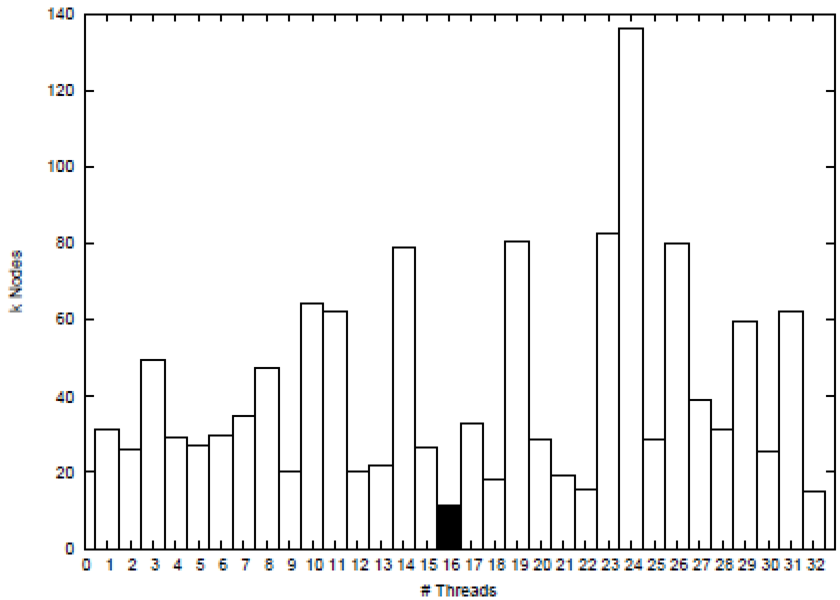




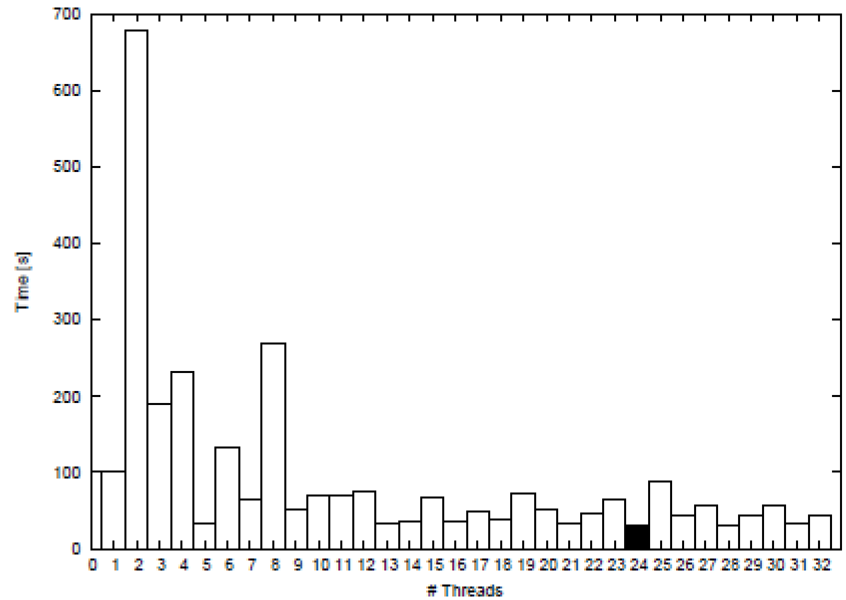


Numbers courtesy of K. Fujisawa

Performance variation due to # of threads



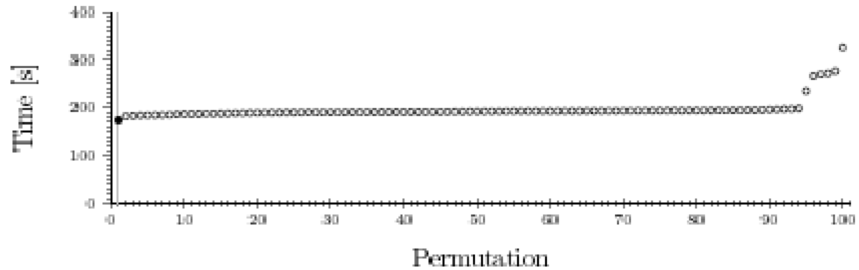
(a) Total number of nodes explored (CPLEX)



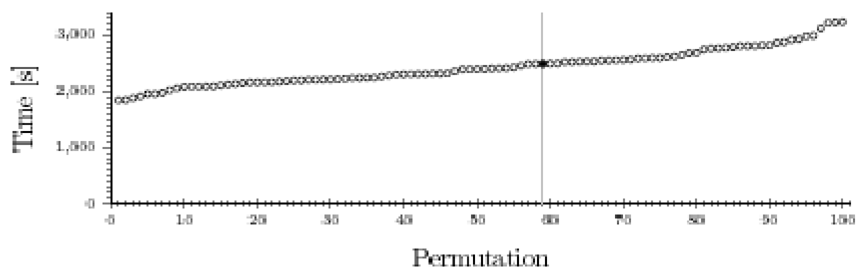
(b) Wall clock solution time (GUROBI)

Fig. 4: Example of performance variability depending on the number of threads. Instance roll3000 on a 32 core computer. Filled bar indicates minimum

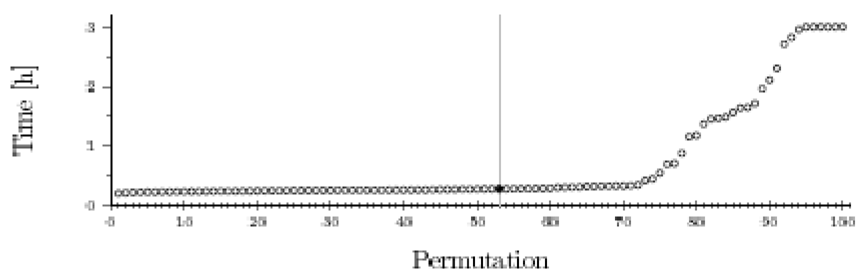
Performance variation due to permutation



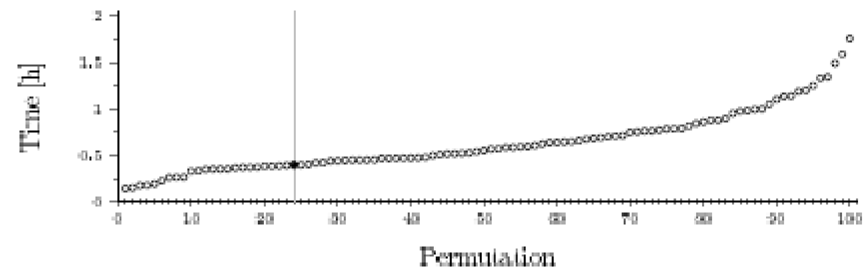
(a) Instance ex9



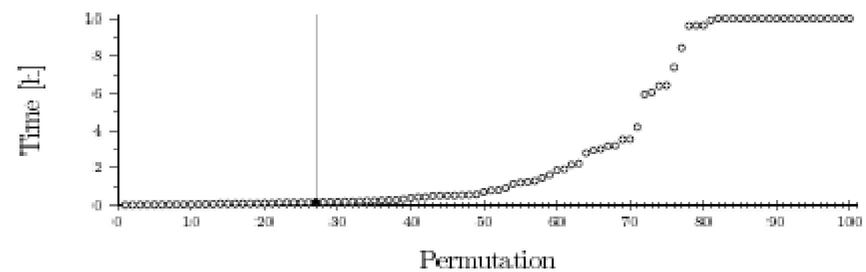
(b) Instance pg5_34



(c) Instance neos13



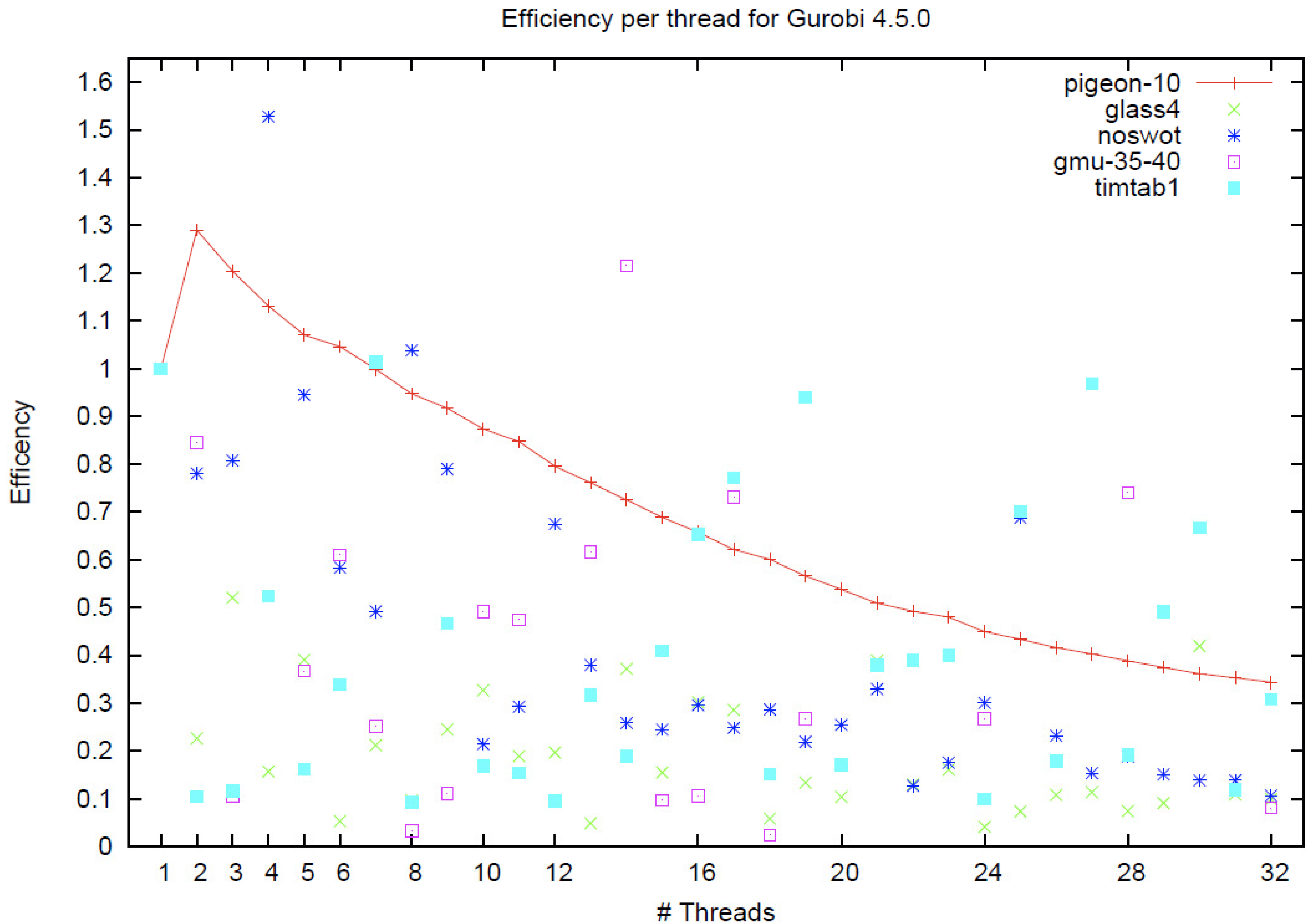
(d) Instance bnatt350



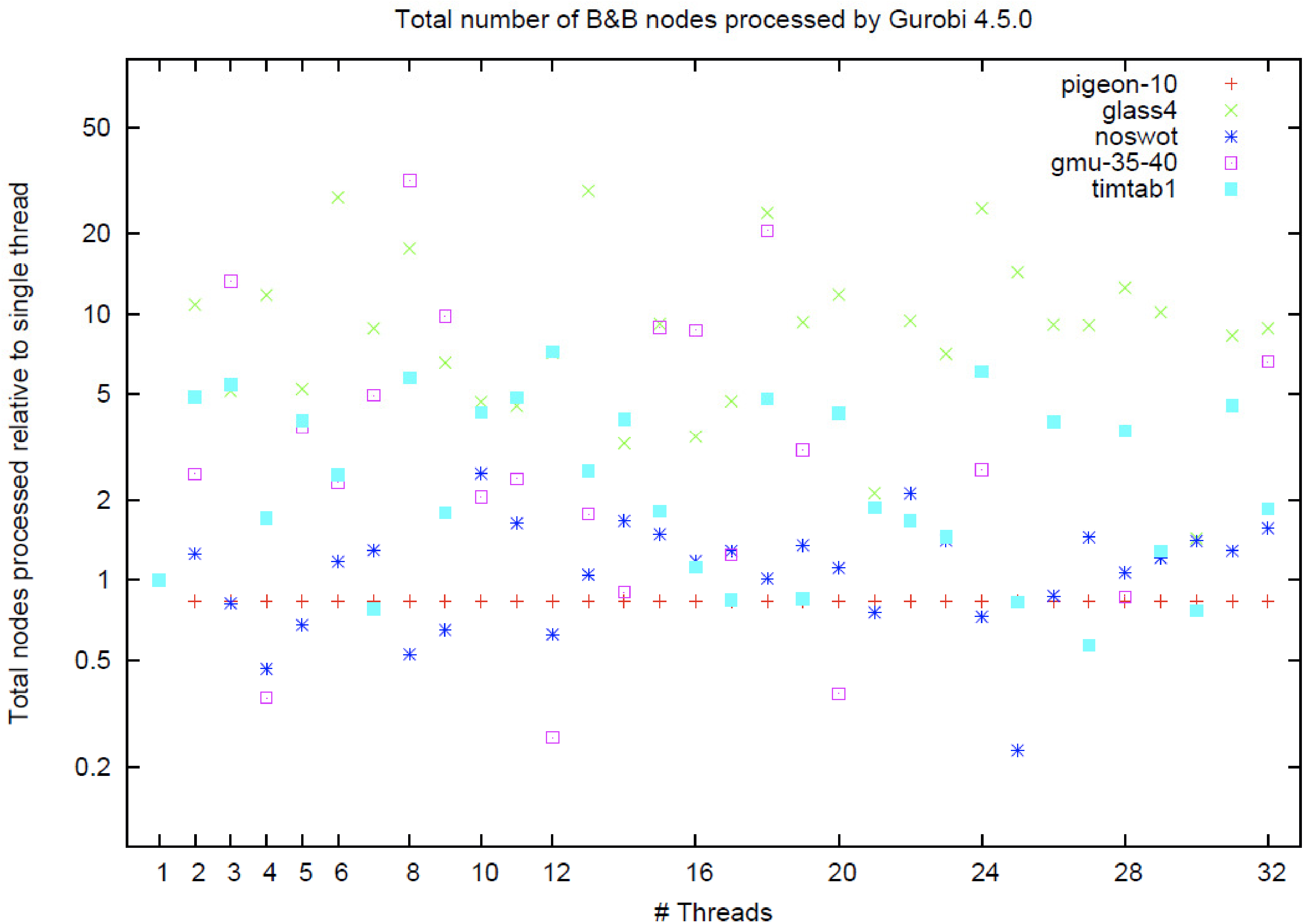
(e) Instance enlight13

Fig. 3: Solution times for 100 permutations

Efficiency of MIP solvers



B&B nodes



Total number of B&B nodes processed by Gurobi 4.5.0
Total nodes processed relative to single thread
50
20
10
5
2
1
0.5
0.2
1
2
3
4
5
6
8
10
12
16
20
24
28
32
Threads
pigeon-10
glass4
noswot
gmu-35-40
timtab1

How to compare MIP solver performance?

There are two components to it:

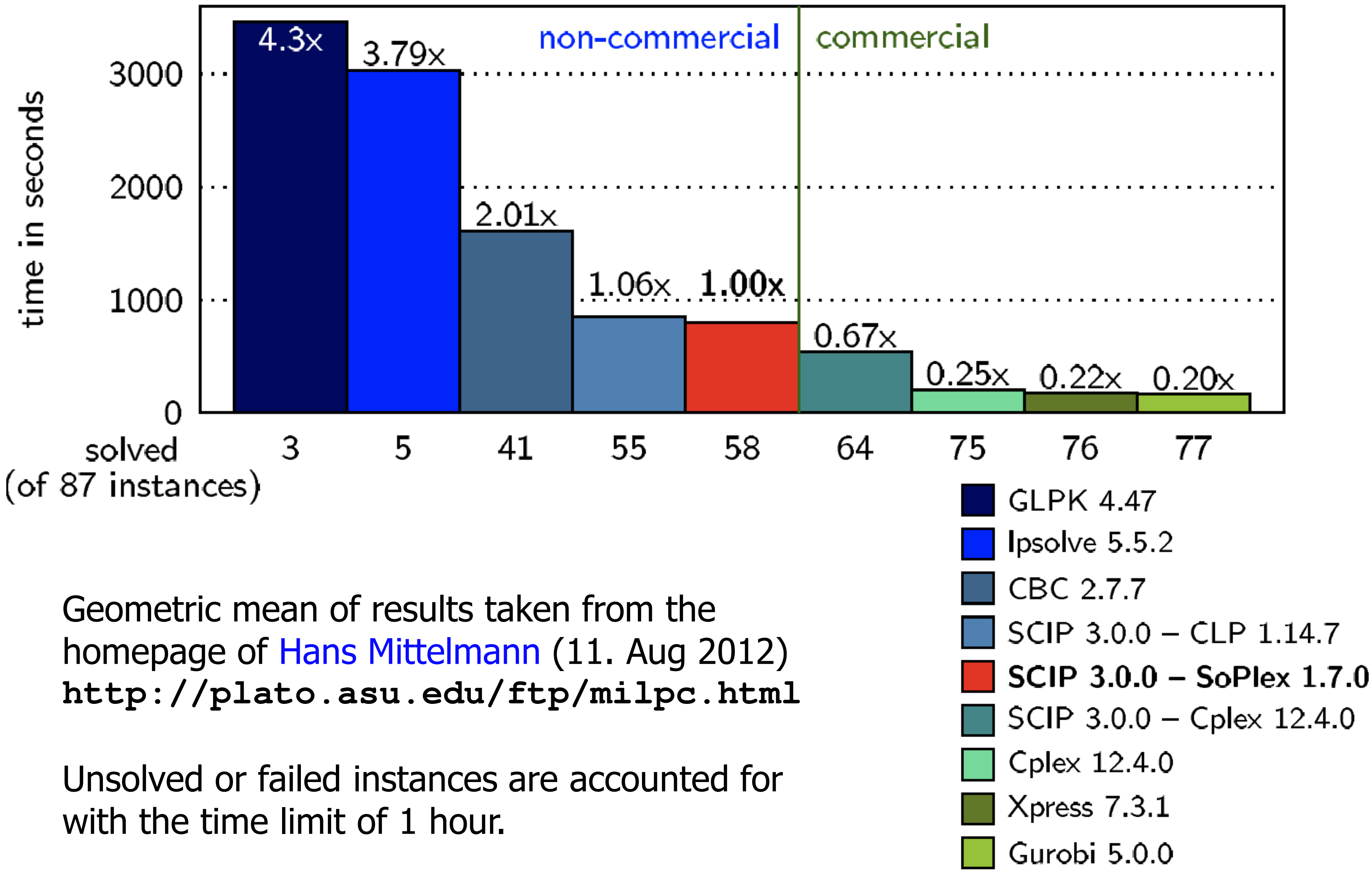
- Ability to solve an instance (within some timeframe)
- Time to solve

You have always to see both numbers!

If two solvers are compared there are two possibilities:

1. Both can solve all instances.
2. There are instances only one solver can solve
3. There are instances both cannot solve

Current MIP Solver performance (1 thread)



Geometric mean of results taken from the homepage of Hans Mittelmann (11. Aug 2012)
`http://plato.asu.edu/ftp/milpc.html`

Unsolved or failed instances are accounted for with the time limit of 1 hour.

Commercial Break

A new open access journal

Mathematical Programming Computation

MPC will publish original research articles covering computational issues in mathematical programming.

Articles report on innovative software, comparative tests, modeling environments, libraries of data, and/or applications.

A main feature of the journal is the inclusion of accompanying software and data with submitted manuscripts. The journal's review process includes the evaluation and testing of the accompanying software. Where possible, the review will aim for verification of reported computational results.

Visit: http://mpc.zib.de and http://springer.com

Editor-in-Chief: William Cook

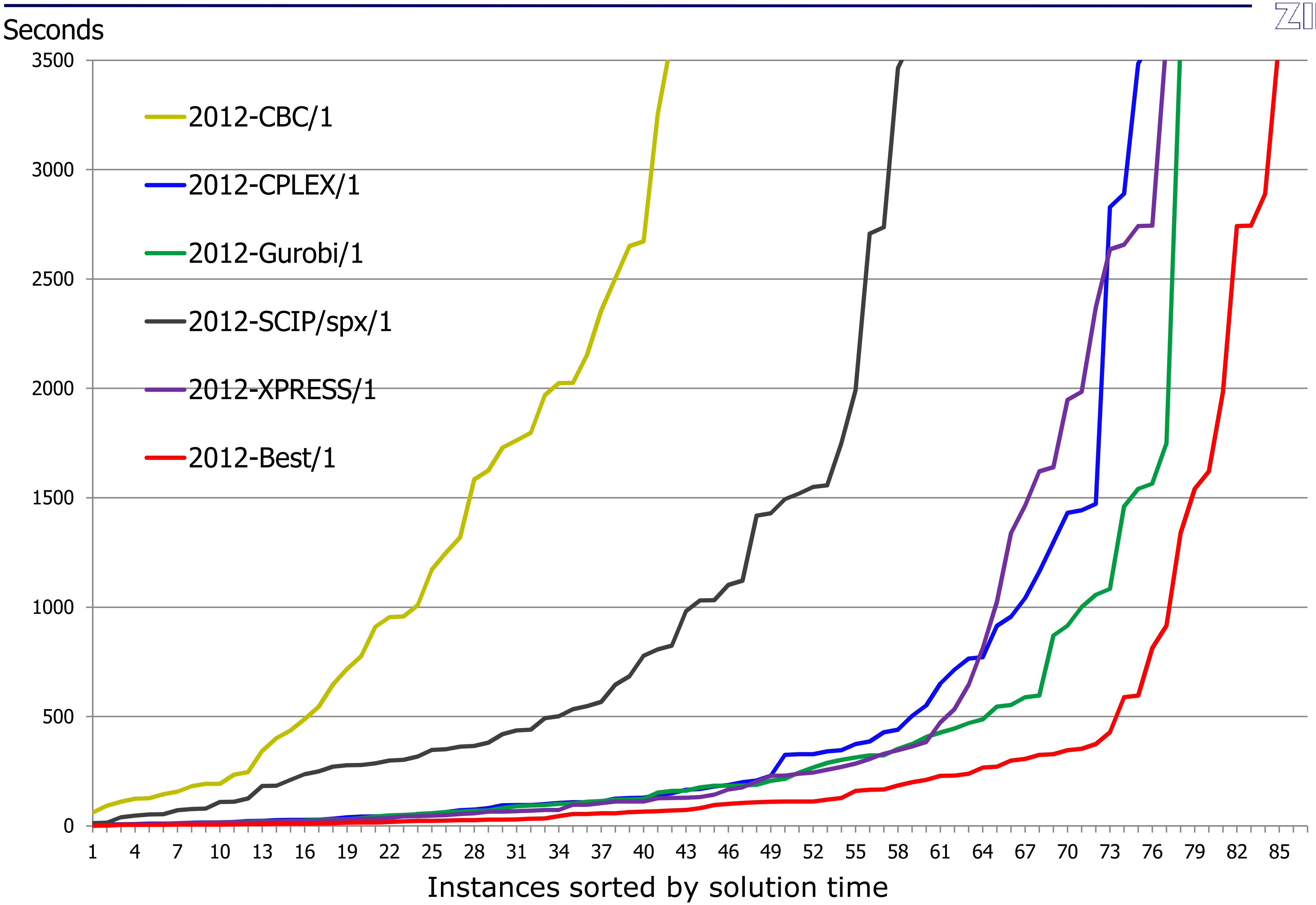
General Editor: Thorsten Koch

Area Editors: D. Bienstock, R. Fourer, A. Goldberg, S. Leyffer, J. Linderoth,
G. Reinelt, K. Toh

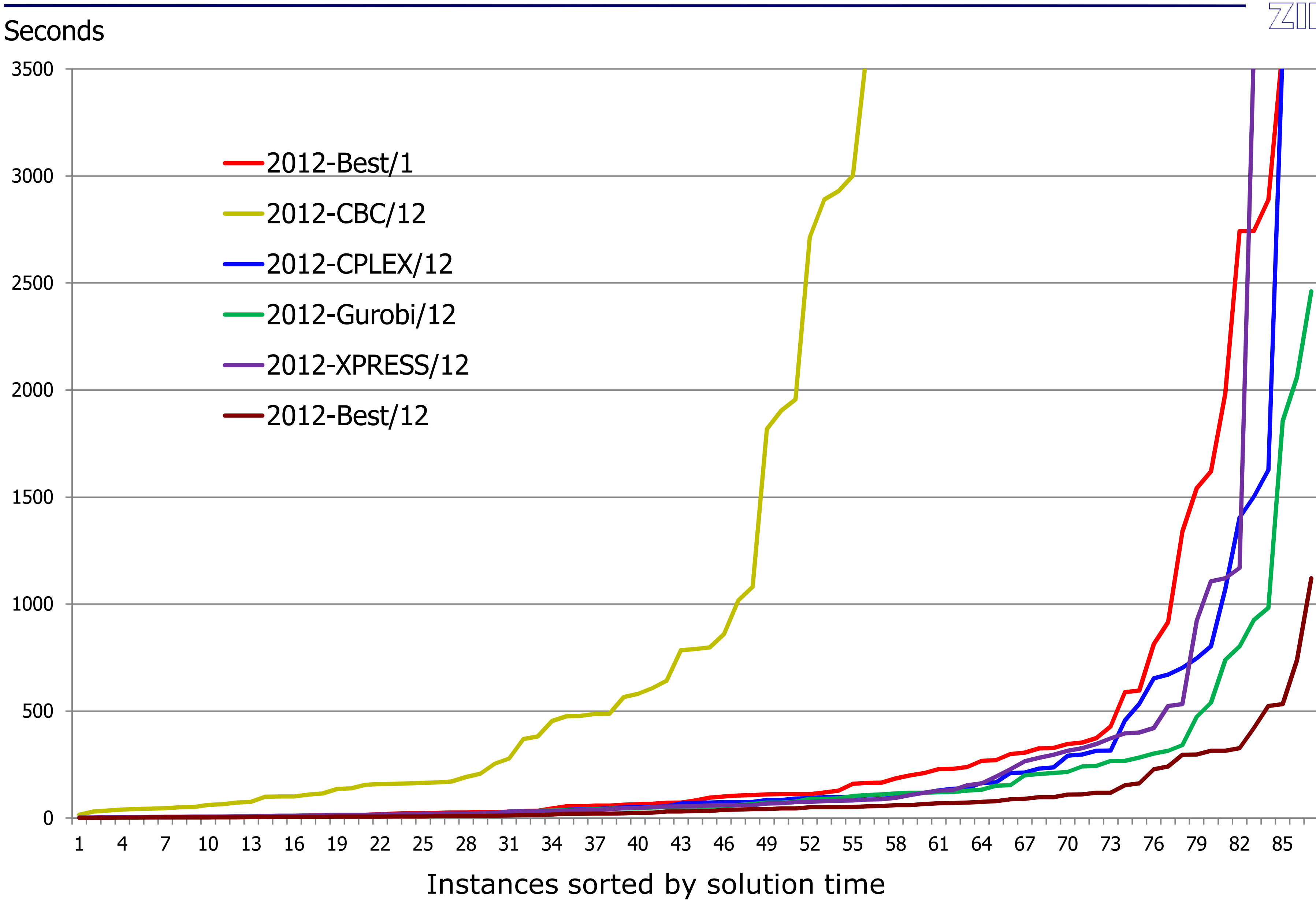
Technical Editors

Tobias Achterberg	Daniel Espinoza	Dominique Orban
David Applegate	Armin Fügenschuh	Marc Pfetsch
Oliver Bastert	Andreas Grothey	Ted Ralphs
Pietro Belotti	Zonghao Gu	Domenico Salvagnin
Hande Y. Benson	William Hart	Martin Schmidt
Andreas Bley	Keld Helsgaun	Mohit Tawarnalani
Brian Borcher	Benjamin Hiller	Stefan Vigerske
Jordi Castro	Leonardo B. Lopes	Richard A. Waltz
Emilie Danna	Todd S. Munson	Stefan Wild

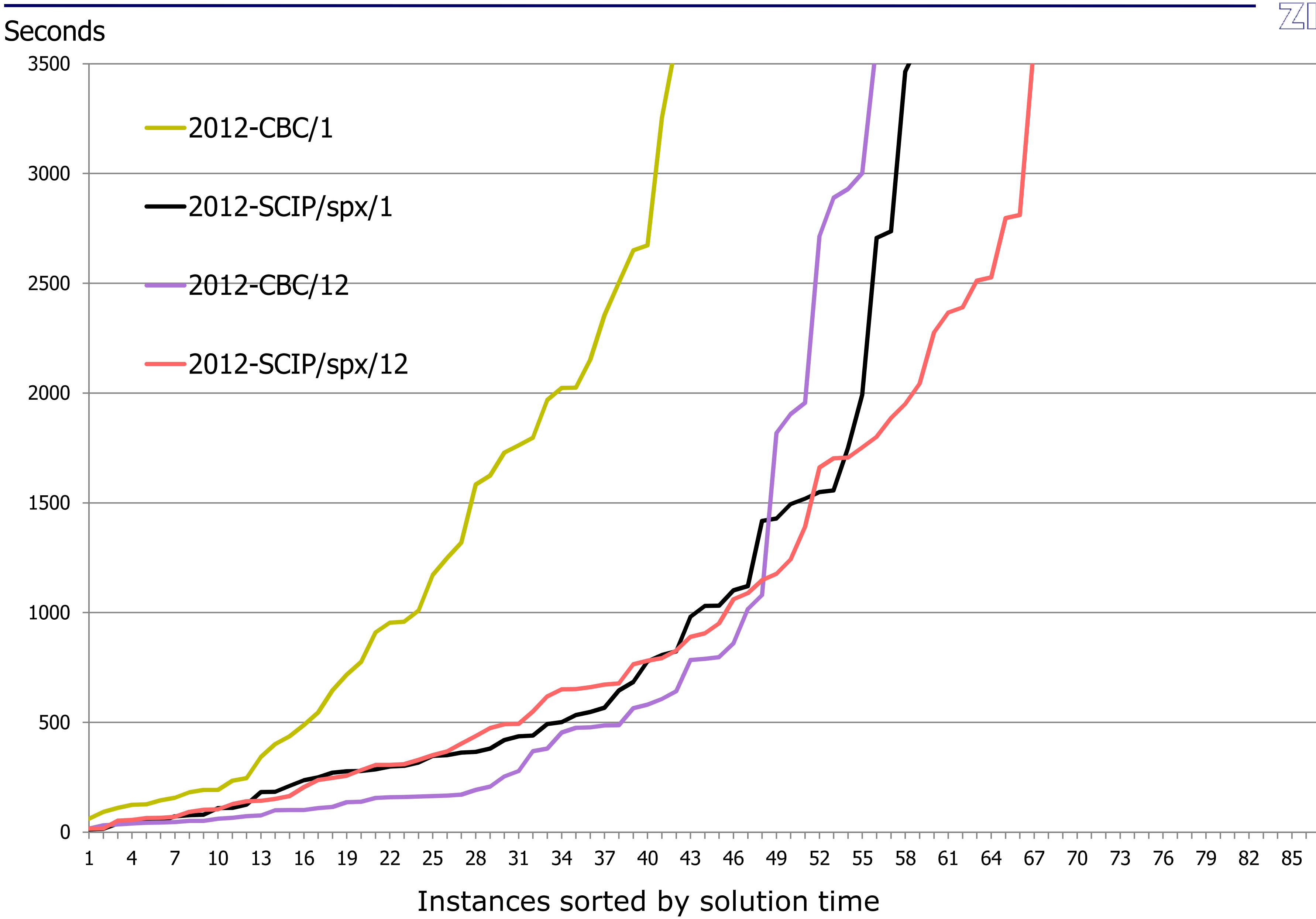
2012 Performance on MIPLIB 2010 / 1 thread



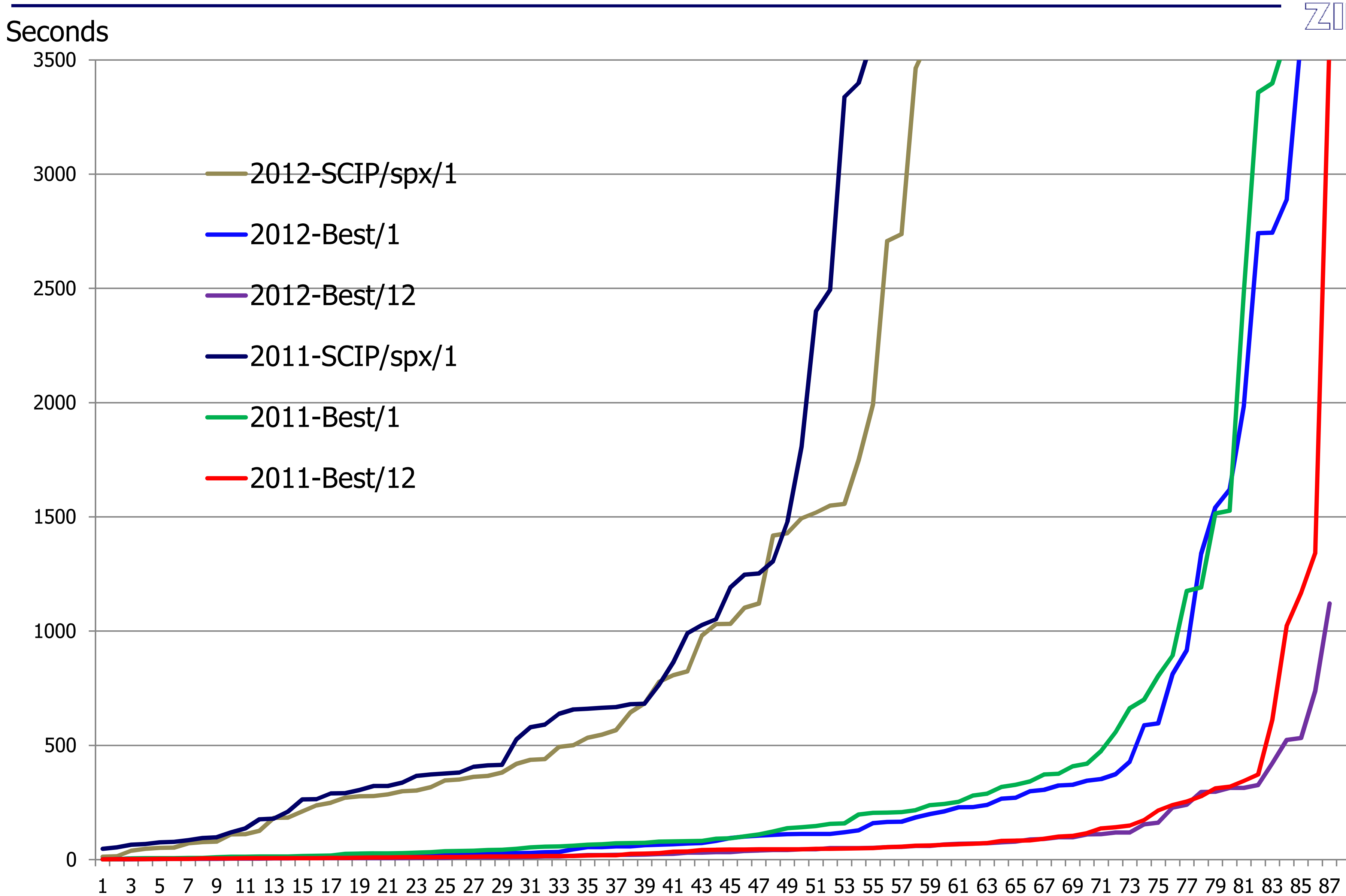
2012 Performance on MIPLIB 2010 / 12 threads



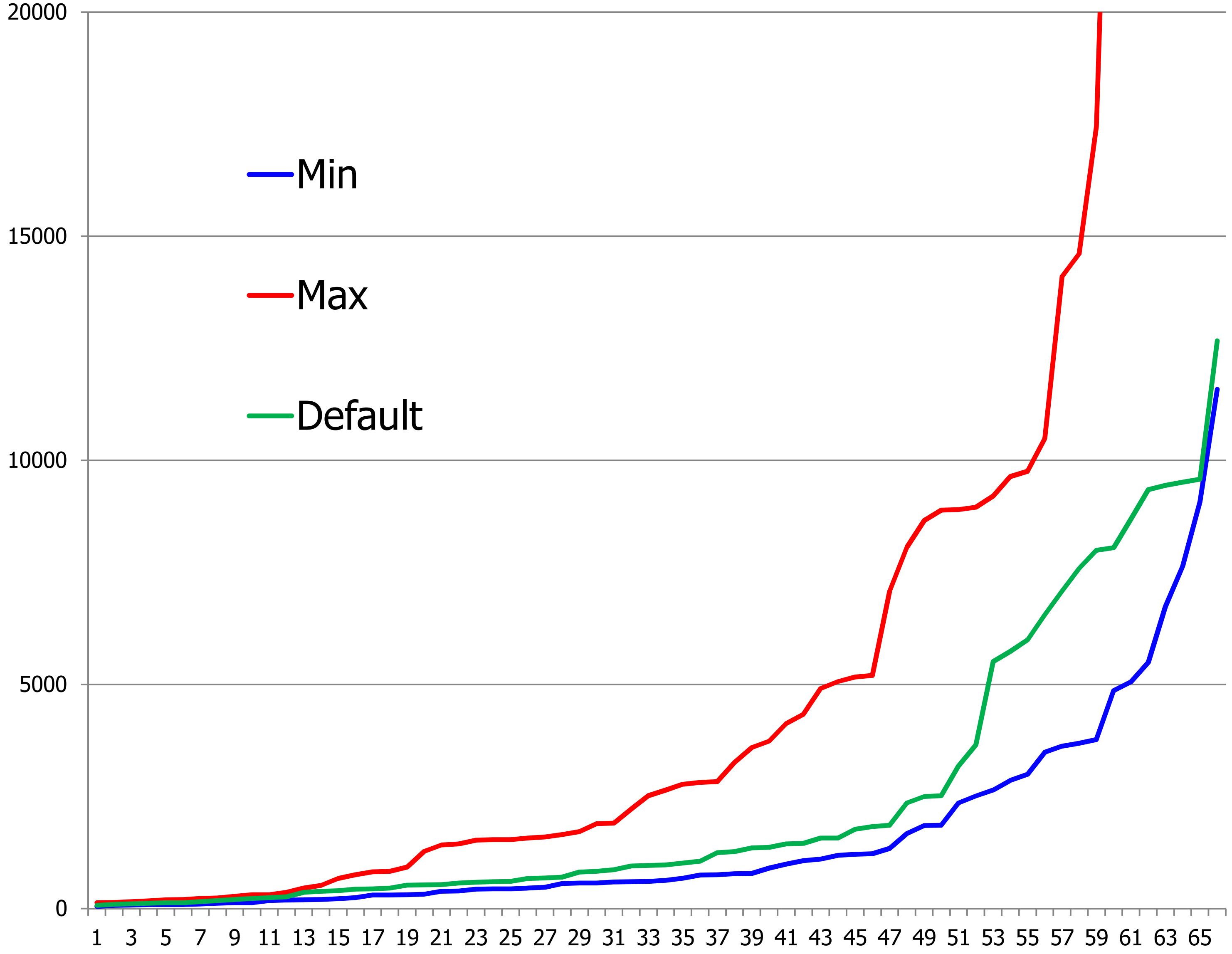
2012 Speedup 1:12 threads CBC/SCIP



MIPLIB Performance 2011:2012



Speedup by permuting

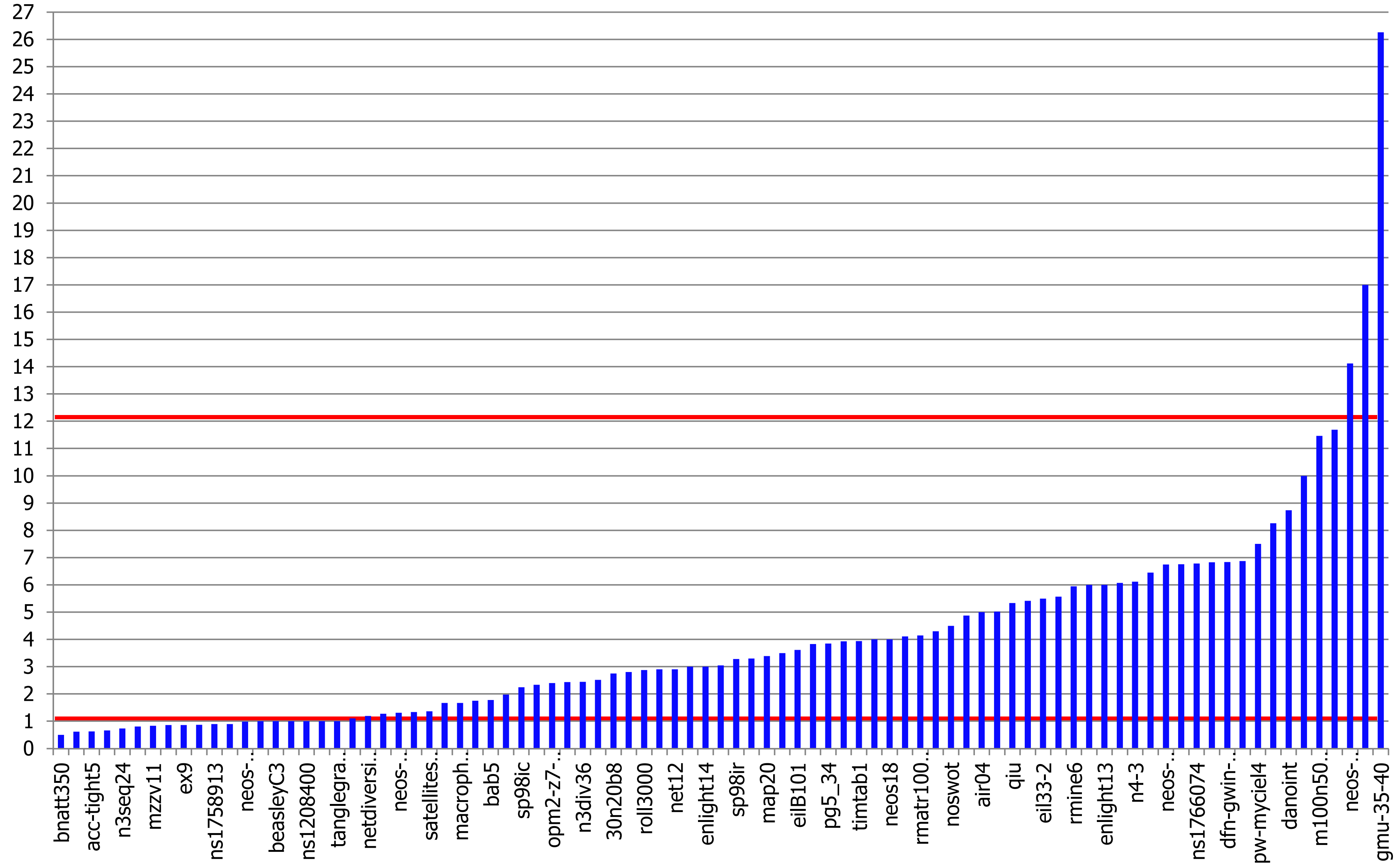


Progress in Numbers

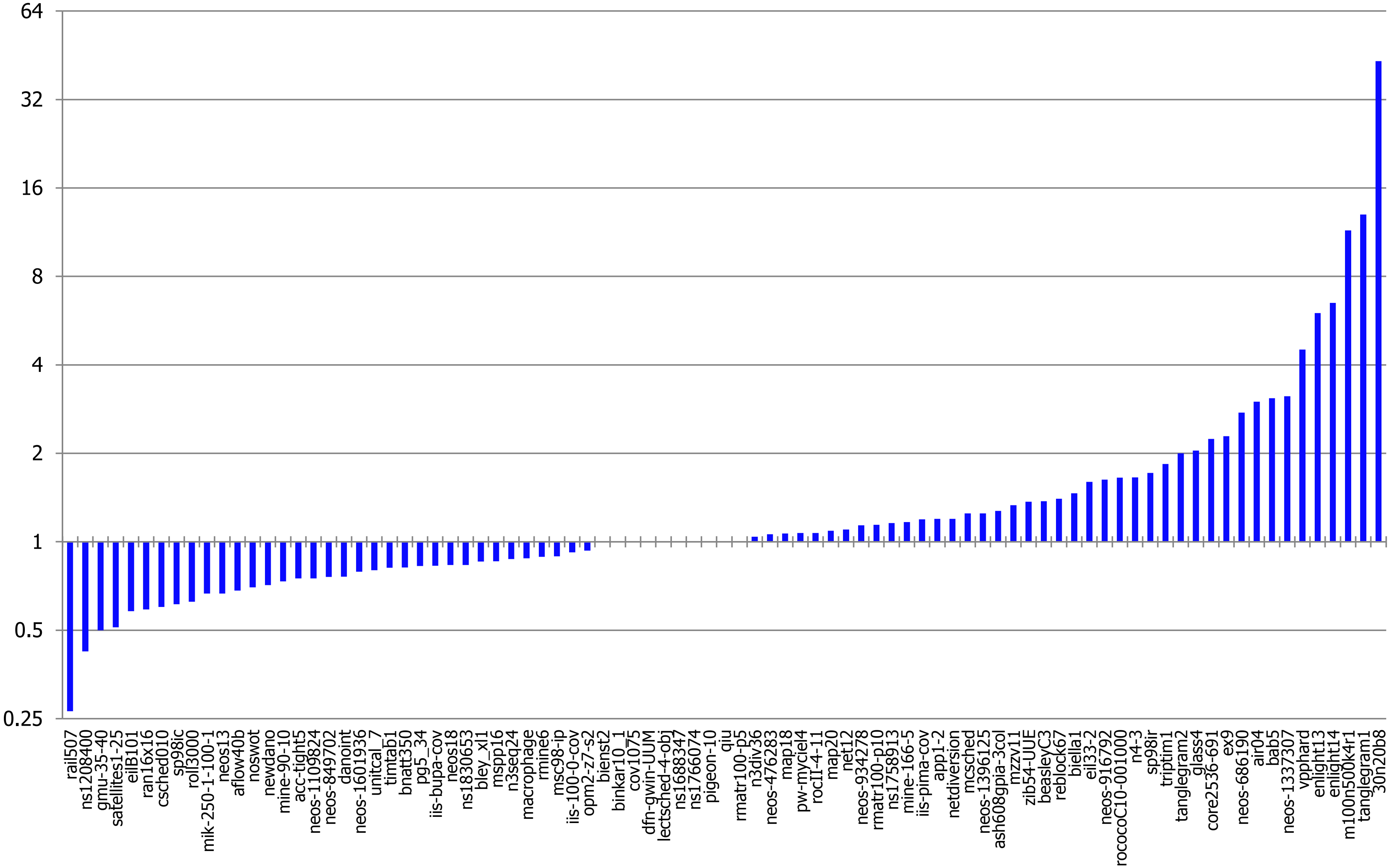
	2011	2012	Speed-up
Best/1 geom. mean time [s]	105	81	**23%**
Best/12 geom. mean time [s]	34	28	**17%**
Best/1 not optimal [#]	4	3	
Best/12 not optimal [#]	1	0	
Speedup 1 to 12 [×]	3.1	2.8	
Max(C,G,X)/Min(C,G,X) 1 thr.	600	259	
Max(C,G,X)/Min(C,G,X) 12 thr.	1138	107	
Best/1 geom. mean nodes [#]	2108	1958	8%
Best/12 geom. mean nodes [#]	2570	2406	7%
Max(C,G,X)/Min(C,G,X) 1 thr.	40,804	32,600	
Max(C,G,X)/Min(C,G,X) 12 thr.	10,499	887,849	

Single solvers have speed-ups up to 42%

2012 Parallel speedup Best 1/12 per instance



Speedup Best/12 2011:2012 per instance



1998 Benchmark Set