Benchmarking a MIP Solver

15 June 2010

Tobias Achterberg
achterberg@de.ibm.com

Computational research

- How can you demonstrate that a new idea, method, algorithm, ... is useful in practice?
  - theoretically prove that the new method is always superior to the current state-of-the-art
  - theoretically prove that the new method has a superior asymptotic behavior

$O(n^{1.9837})$ vs. $O(n^2)$

  - benchmark against state-of-the-art solver on appropriate problem instances

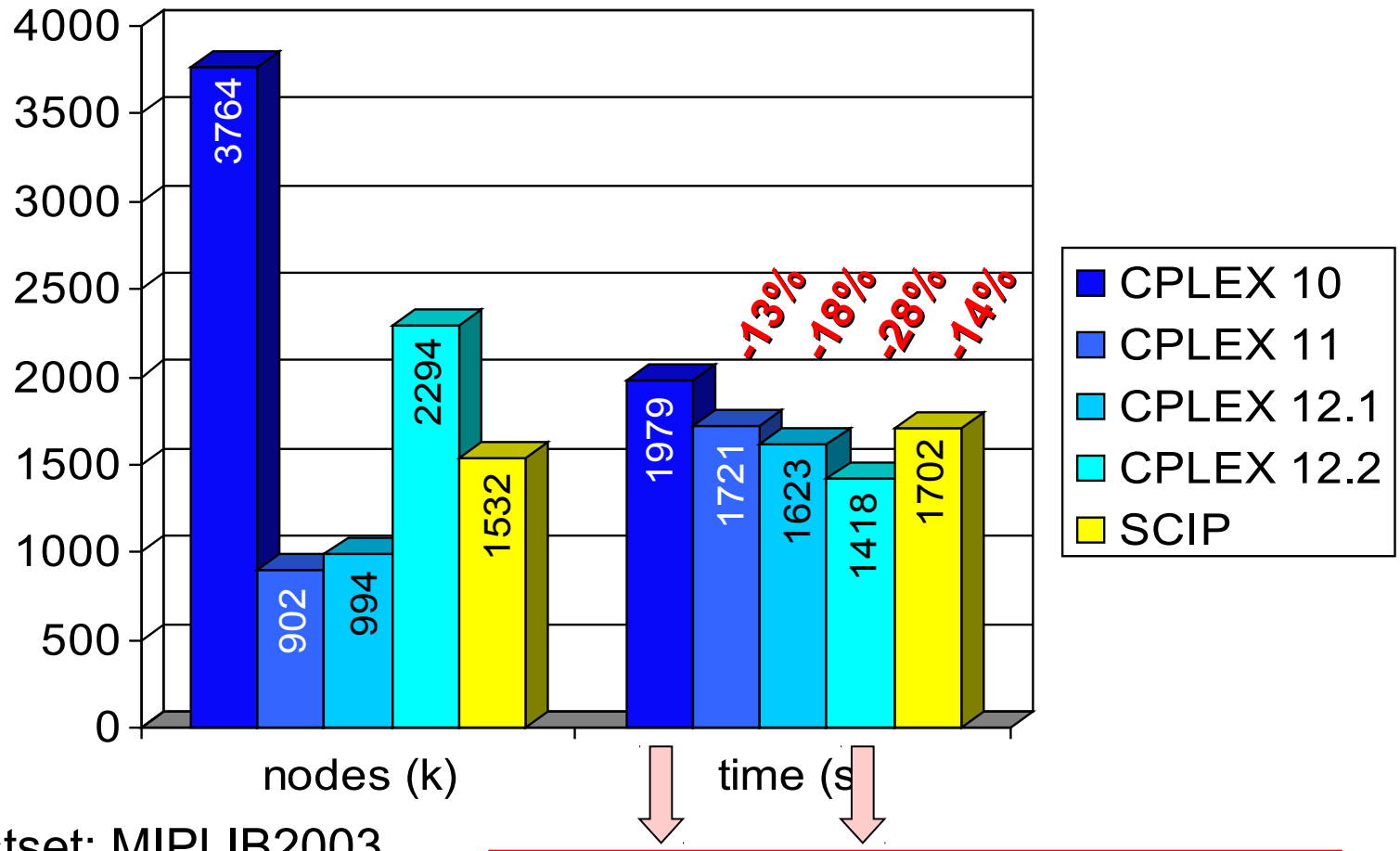
Main issues for researchers

- Implementing algorithms is very laborious
- Benchmarking is very time-consuming
  - collect and carefully select models
  - run tests
  - interpret results, adjust algorithms, rerun tests
- Result of all this work: one table with numbers
  - looks much less impressive than a nice theorem
  - everyone will complain about your test methodology
  - reviewers tend to ask for *even more* detailed results
    - easy to suggest experiments, but hard to conduct them

Experimental setup

- Compare 5 different solvers
  - CPLEX 10.0 (January 2006)
  - CPLEX 11.0 (October 2007)
  - CPLEX 12.1 (June 2009)
  - CPLEX 12.2 (June 2010)
  - SCIP 1.2.0 (September 2009)
    - using CPLEX 12.2 as LP solver
- Intel Xeon X5260 @ 3.33 GHz
  - 4 cores, 6 MB cache, 16 GB RAM
  - if compared to SCIP use only 1 thread
- Time limit 3600 seconds, memory limit 6 GB

First experiment on MIPLIB 2003



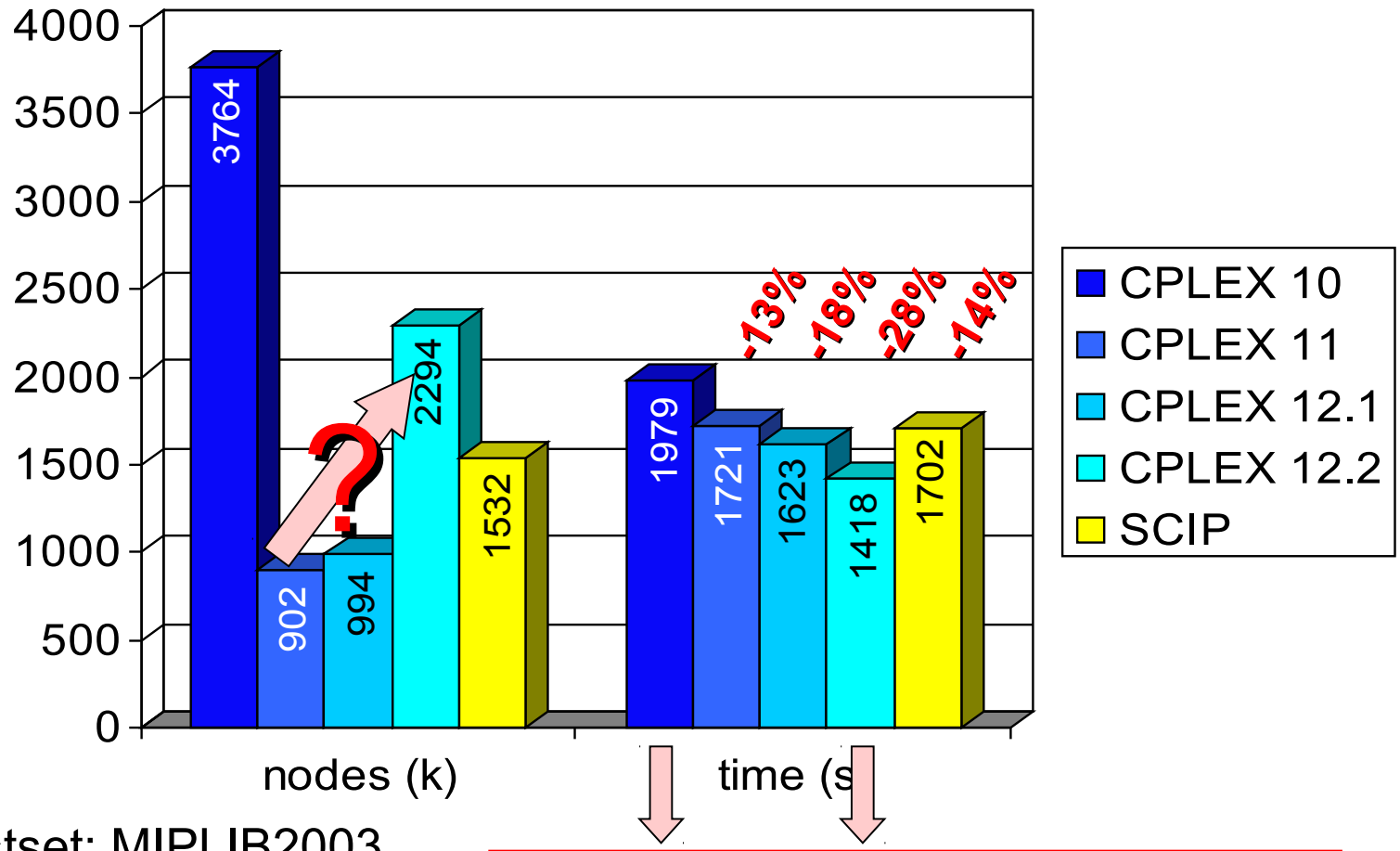
- Testset: MIPLIB2003
  - 60 models
  - time limit 3600 seconds

Time reduced by only 28% from CPLEX 10 to CPLEX 12.2

What is wrong?

- 22 models hit time limit of 3600 seconds **for all solvers**
  - large constant offset in arithmetic means of time
  - shifts all time ratios between solvers towards 1.0

First experiment on MIPLIB 2003



- Testset: MIPLIB2003
  - 60 models
  - time limit 3600 seconds

Time reduced by only 28% from CPLEX 10 to CPLEX 12.2

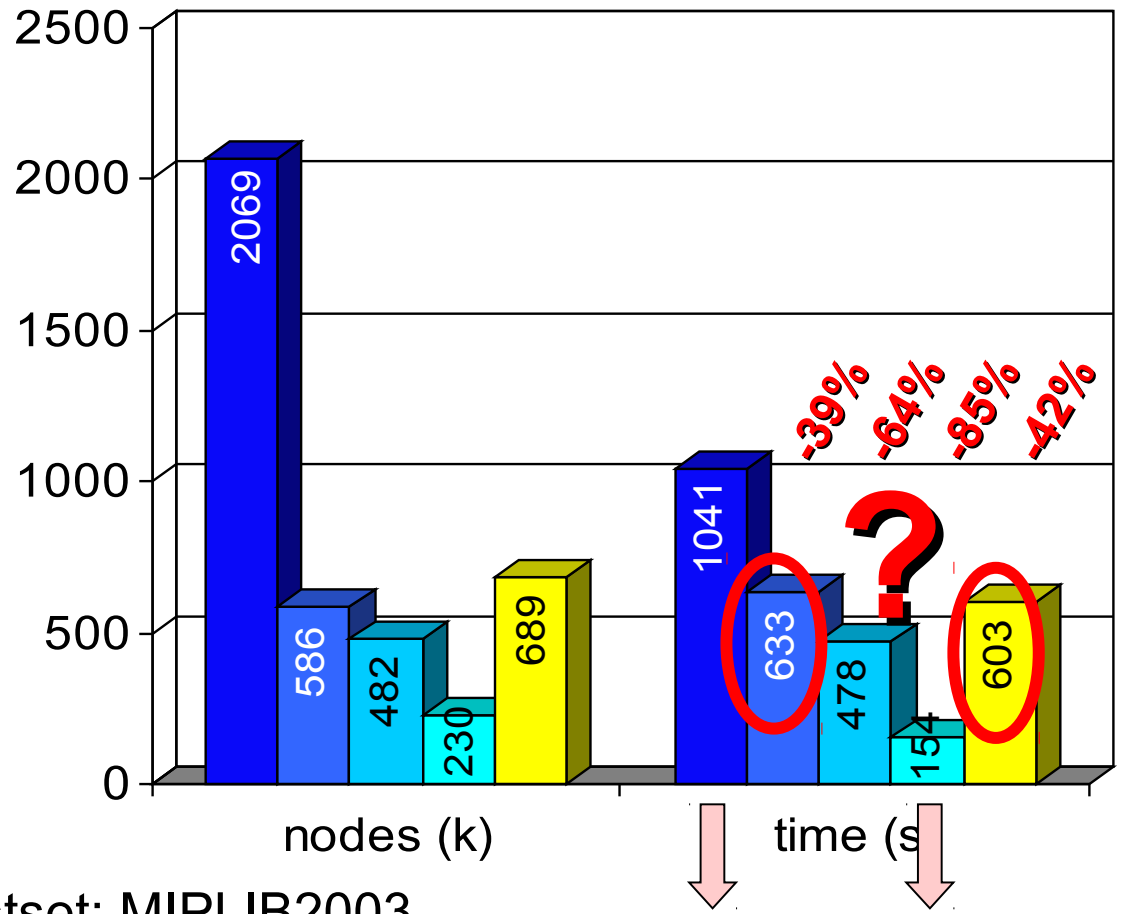
What is wrong?

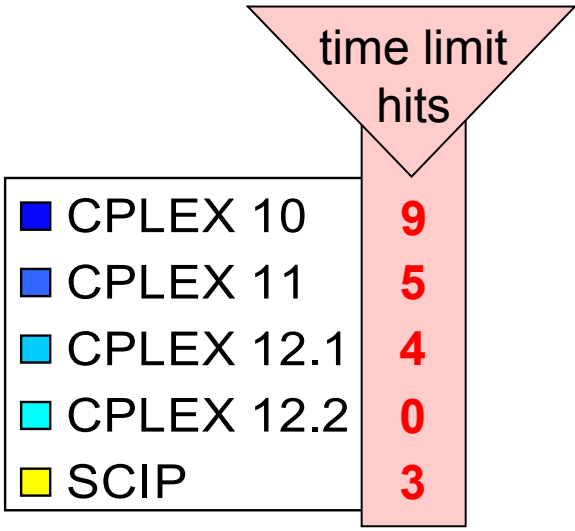
- 22 models hit time limit of 3600 seconds **for all solvers**
  - large constant offset in arithmetic means of time
  - shifts all time ratios between solvers towards 1.0
- If time limit is hit, node counts are not very meaningful
  - increase in the number of nodes just means that **CPLEX 12.2 has a higher node throughput than 11 and 12.1**

Conclusion

Remove models that none of the solvers can solve from the test set!

MIPLIB 2003 – removed unsolved models





- Testset: MIPLIB2003
  - 38 models
  - time limit 3600 seconds

6.8x speed-up from CPLEX 10 to CPLEX 12.2

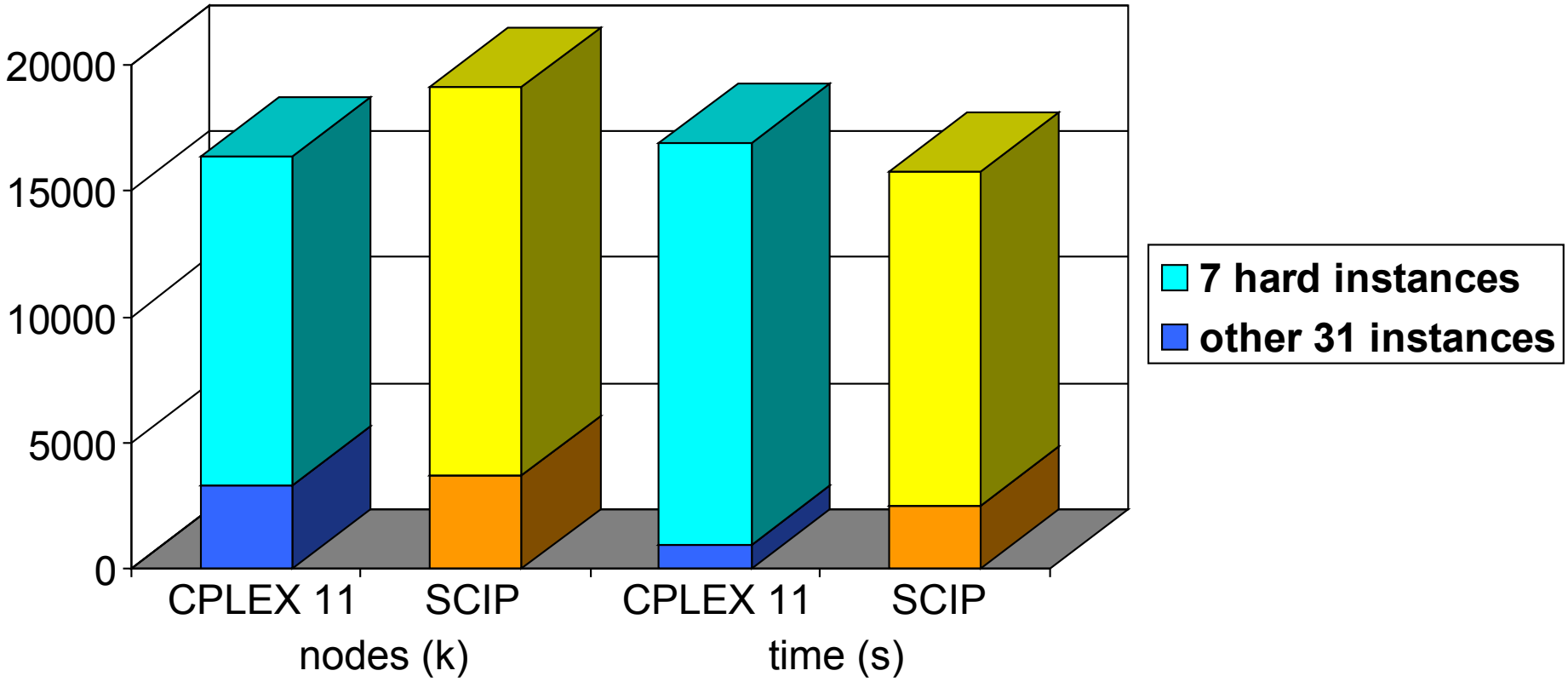
Models sorted by max time

	Model	CPLEX 11	SCIP	sum(CPX)	sum(SCIP)
31	opt1217	0.1	0.3		
	...	..	...	...	...
	mas74	196.1	770.9	975.9	2503.3
7	aflow40b	2005.3	1459.9	2981.2	3963.2
	timtab1	2558.5	562.1	5539.7	4525.3
	tr12-30	397.8	3394.3	5937.5	7919.6
	harp2	128.9	3600.0	6066.4	11519.6
	noswot	3600.0	360.6	9666.4	11880.2
	arki001	3600.0	1864.5	13266.4	13744.7
	net12	3600.0	1973.7	16866.4	15718.4

CPLEX 11 is 2.5x faster!

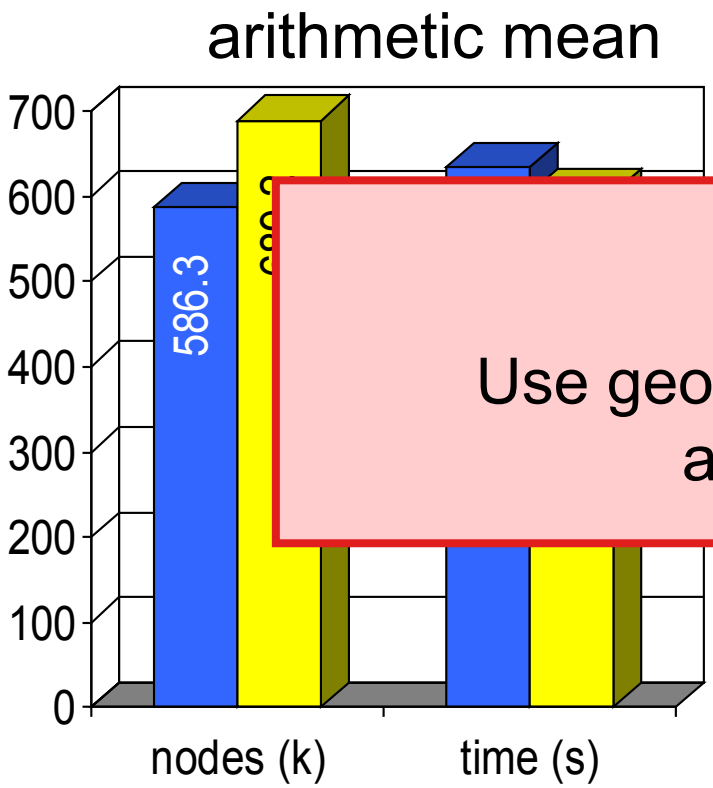
- Only 7 models relevant for arithmetic mean!
- Better luck on 5 of them for SCIP

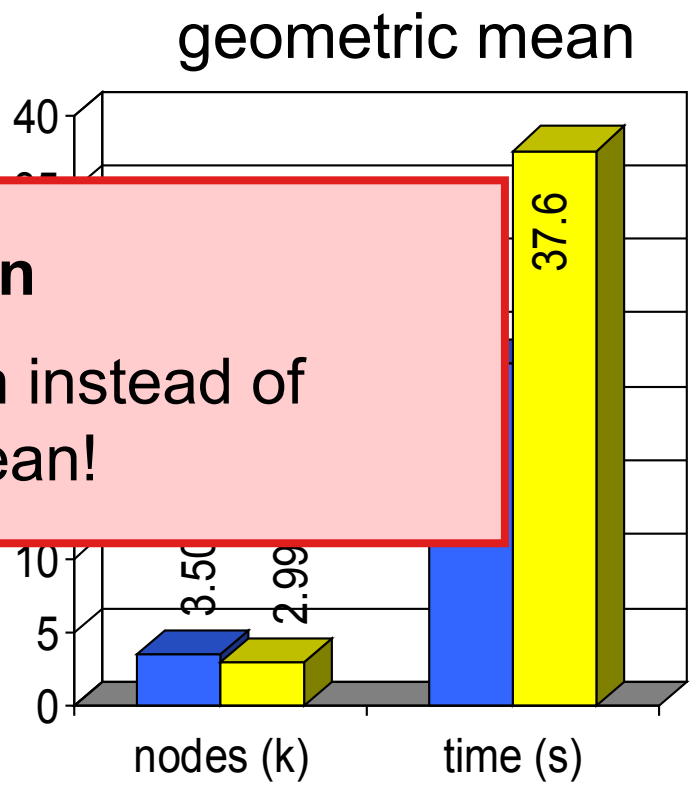
Hard models dominate totals



- 7 out of 38 models dominate the totals
- Better: geometric mean

Arithmetic versus geometric mean





Conclusion

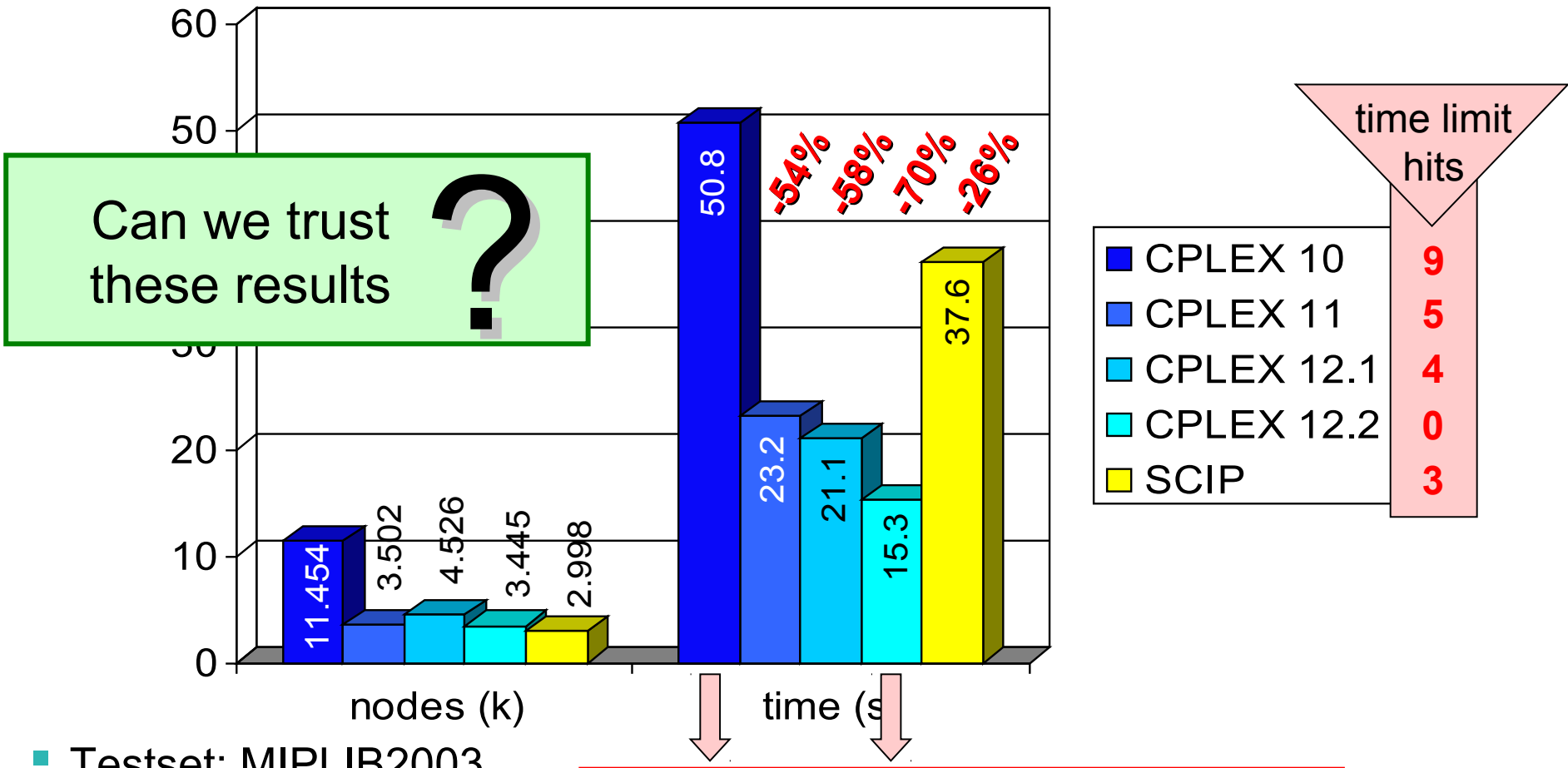
Use geometric mean instead of arithmetic mean!

ratios: SCIP/CPLEX11	nodes	time
arithmetic mean	1.18	0.95
geometric mean	0.86	1.62

SCIP faster

CPLEX faster

Final MIPLIB 2003 results: geometric means

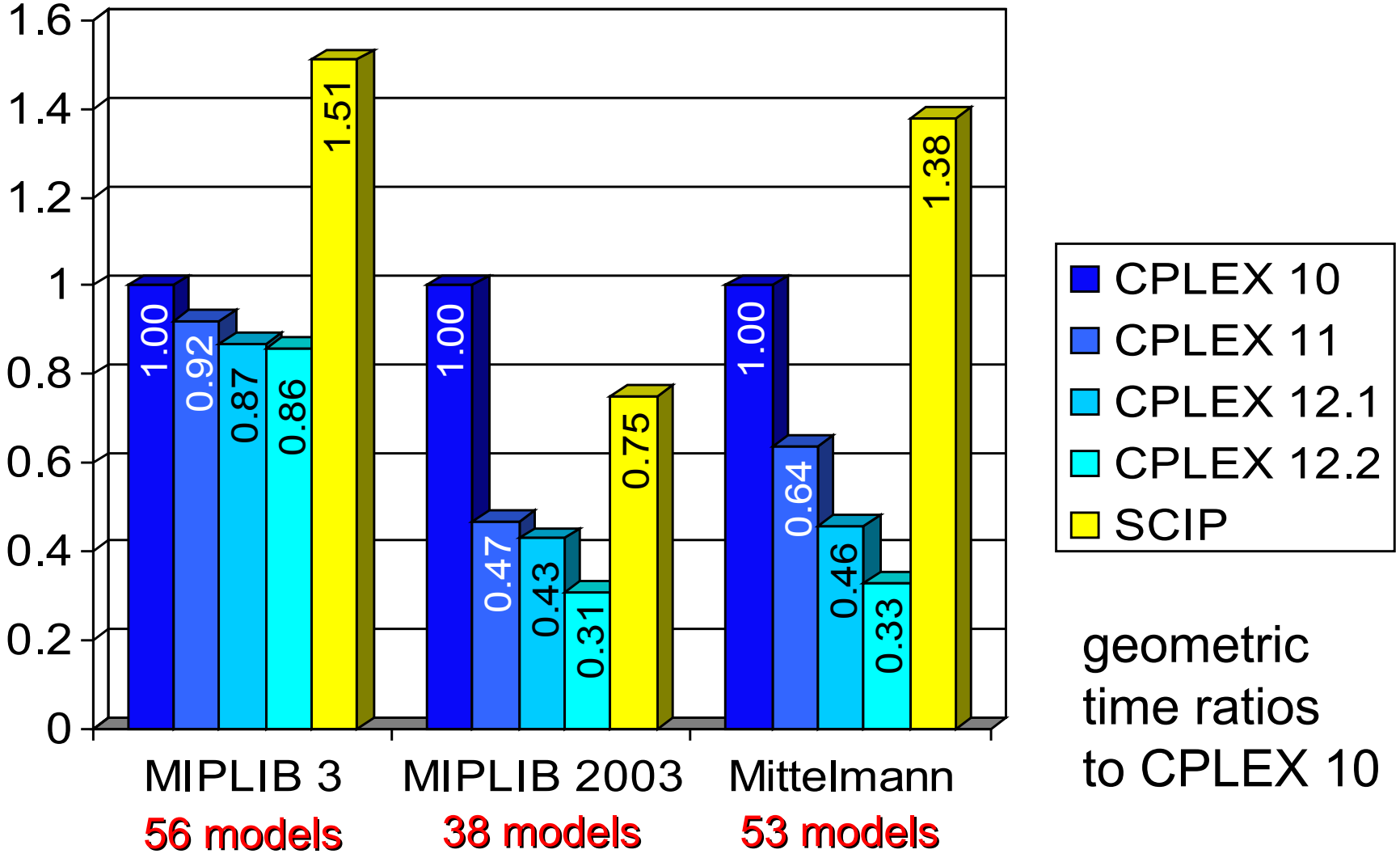


Can we trust these results ?

- Testset: MIPLIB2003
  - 38 models
  - time limit 3600 seconds

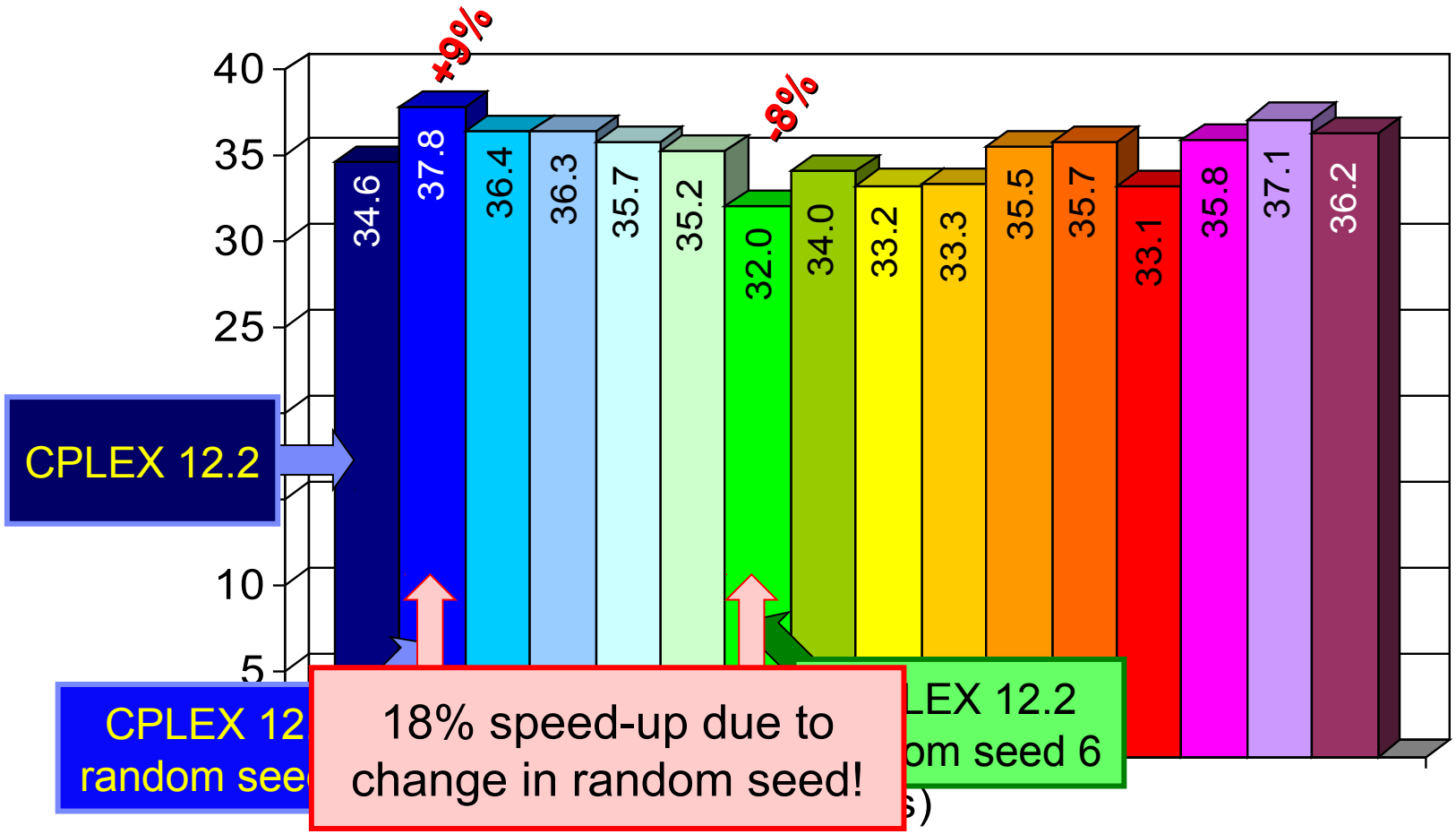
3.3x speed-up from CPLEX 10 to CPLEX 12.2

Variability across test sets



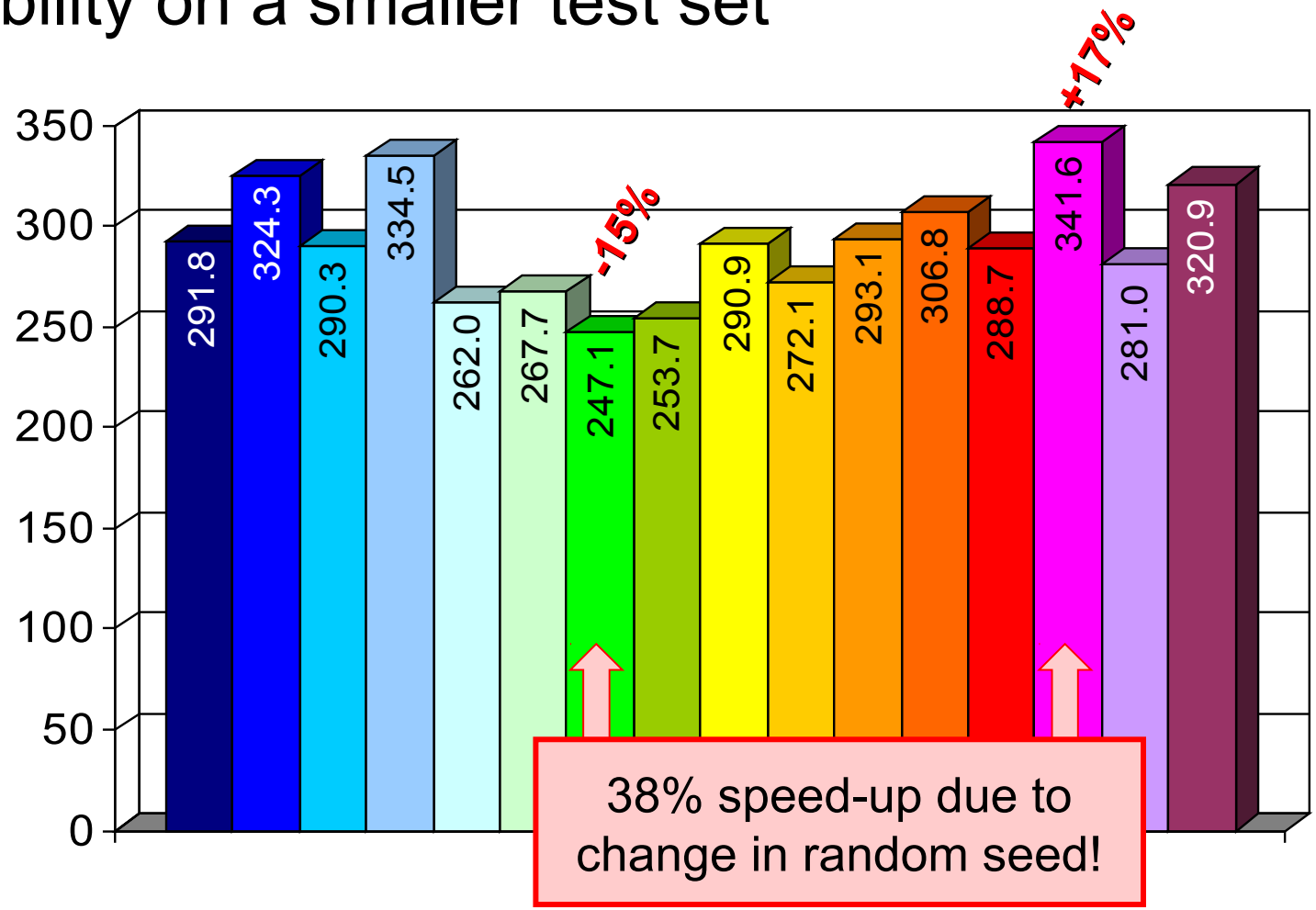
geometric time ratios to CPLEX 10

Some more solvers ...



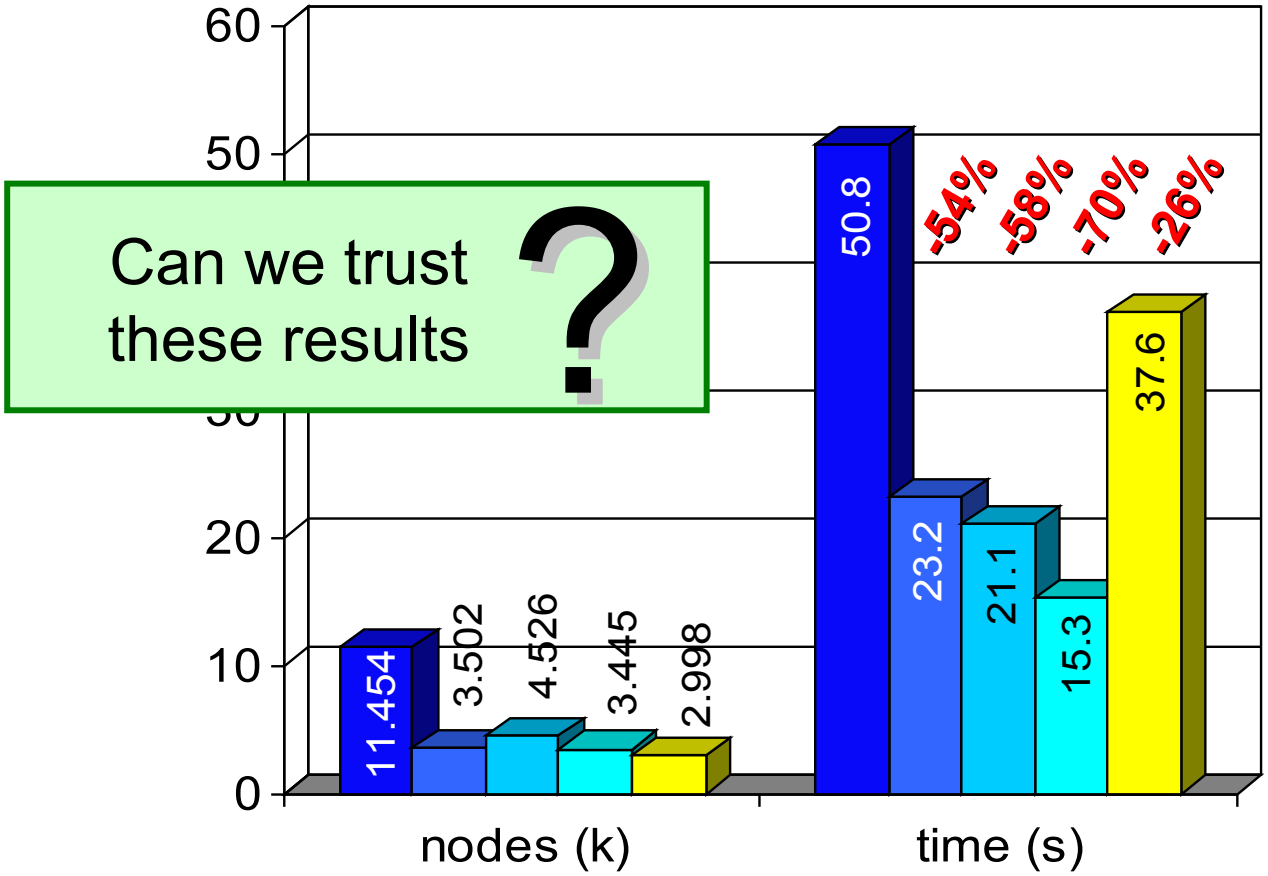
- Testset: MIPLIB 3, MIPLIB 2003, Mittelmann
  - 76 models with max(t) ≥ 1 and min(t) < 3600

Variability on a smaller test set



- Testset: MIPLIB 3, MIPLIB 2003, Mittelmann
  - 29 models with max(t) ≥ 100 and min(t) < 3600

Final MIPLIB 2003 results: geometric means



Can we trust these results ?

	time limit hits
CPLEX 10	9
CPLEX 11	5
CPLEX 12.1	4
CPLEX 12.2	0
SCIP	3

- Testset: MIPLIB2003
  - 38 models
  - time limit 3600 seconds

test set too small!

Dealing with variability

- Many MIP models are highly variable
  - seemingly performance neutral changes to the code significantly affect the solving time
  - luck on a few models dominates benchmark result
- Distinguish between noise and true performance change
  - inferential statistics
    - Student‘s t-test
    - confidence intervals
  - truncated mean values
    - leave out n best and n worst results from mean values
  - **LARGER TEST SETS!**
    - luck averages out
    - at least 100 models, better many more!

Summary

- Use geometric instead of arithmetic means
  - avoid dominance of a few hard models
- Consider other statistics
  - inferential methods like Student‘s t-test to provide confidence intervals
  - truncated mean values to reduce effect of outliers
  - number of models that are solved faster and slower by x%
    - performance profile
- **Use large test sets**
  - reduce influence of random artifacts
  - at least 100 models
  - avoid overrepresentation of single model class

Additional pitfalls

- Use same abort criteria for all solvers
  - relative gap calculated differently in solvers ⇒ use mipgap = 0
- Check for inconsistencies in the solver output
  - exclude instances with different results
  - list number of wrong answers as additional benchmark indicator
- Failed runs due to bugs
  - disregard instance for the mean value calculation
  - list number of failed runs as additional benchmark indicator
- How to account for hitting the time limit?
  - pretend that instance was solved at this moment
  - list number of time limit hits as additional benchmark indicator
- How to account for hitting the memory limit?
  - increase solving time to time limit
  - scale nodes and iterations accordingly

Finally...

- Look at the individual numbers
  - if something is suspicious, rerun the instance
  - background processes corrupt timings!
    - dual core CPU: up to 40% deterioration in time
    - hyperthreading CPU: much worse
    - Intel Nehalem: faster single core runs if other cores idle
  - seemingly identical machines may perform differently
    - different RAM modules (4x2 GB instead of 2x4 GB)
    - RAM modules located in different slots
    - different cooling situation in server room

Comparison of CPLEX 12.1 and 12.2 – MIP sequential

sup(time)	#	time limit hits	faster	slower	time	nodes
[0,1)	1025	—	38	52	1.01	0.89
[1,10)	542	—	158	216	1.02	0.86
[10,100)	406	—	201	147	0.89	0.89
[100,1k)	308	—		0	0.66	0.85
[1k,10k)	228	—	144	61	0.61	0.76
[1,10k}	1590	87 / 19	764	532	0.77	0.80
[10,10k}	1048	87 / 19	606	316	0.67	0.77
[100,10k}	642	87 / 19	405	169	0.55	0.71
[1k,10k}	334	87 / 19	227	79	0.47	0.60

1.30x speed-up

2.13x speed-up

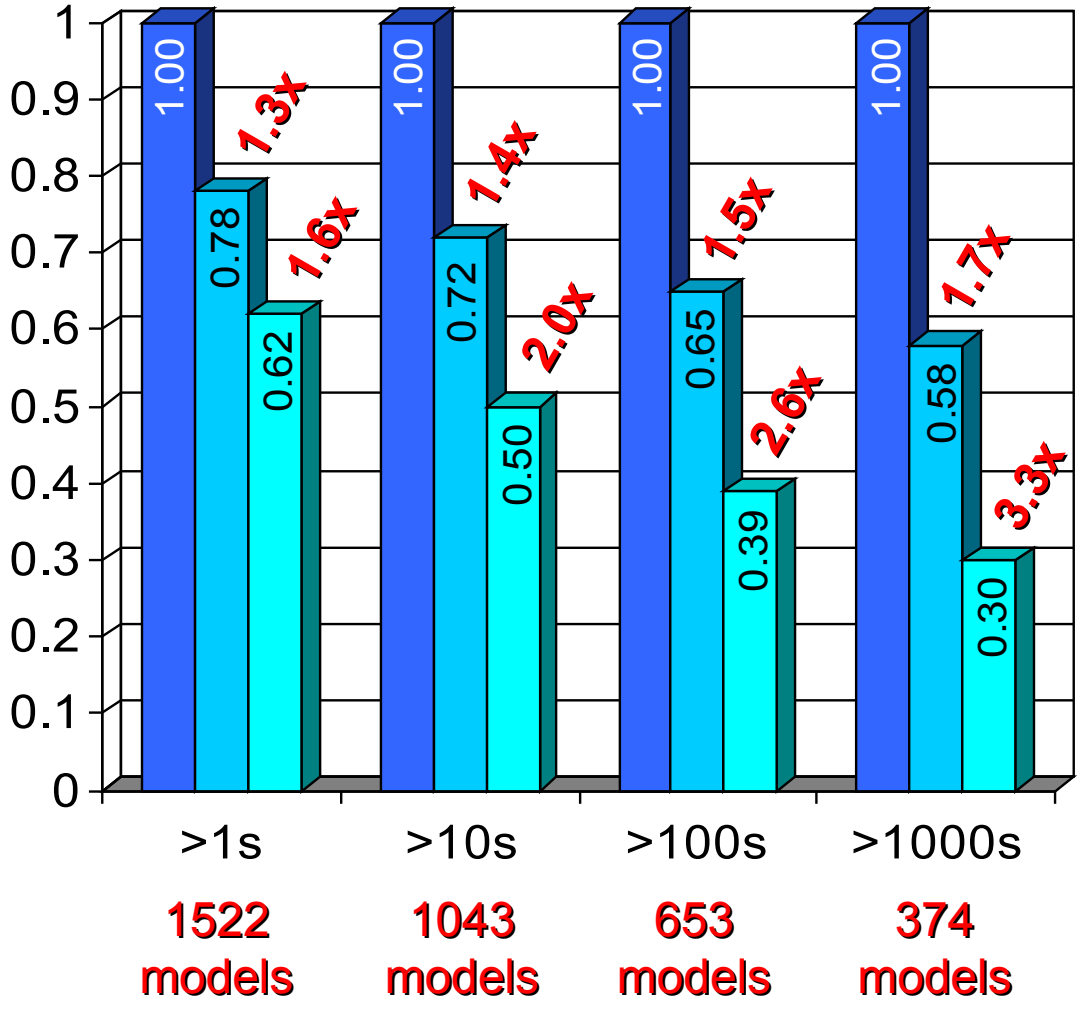
Comparison of CPLEX 12.1 and 12.2 – MIP 4 threads

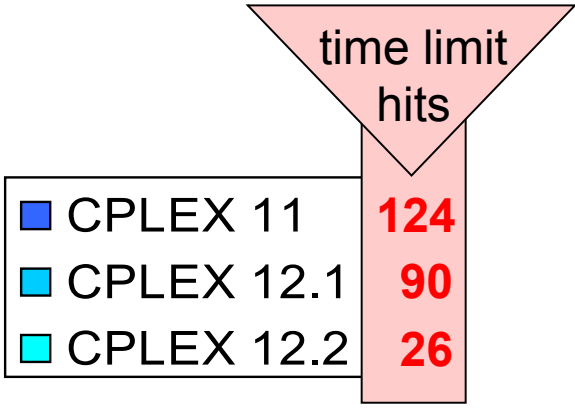
sup(time)	#	time limit hits	faster	slower	time	nodes
[0,1)	1060	—	44	78	1.01	0.83
[1,10)	594	—	198	242	1.01	0.82
[10,100)	389	—	240	107	0.73	0.82
[100,1k)	308	—		4	0.56	0.87
[1k,10k)	207	—	148	42	0.43	0.64
[1,10k}	1627	97 / 32	891	492	0.68	0.78
[10,10k}	1033	97 / 32	693	250	0.54	0.76
[100,10k}	644	97 / 32	453	143	0.45	0.73
[1k,10k}	336	97 / 32	241	69	0.37	0.62

1.47x speed-up

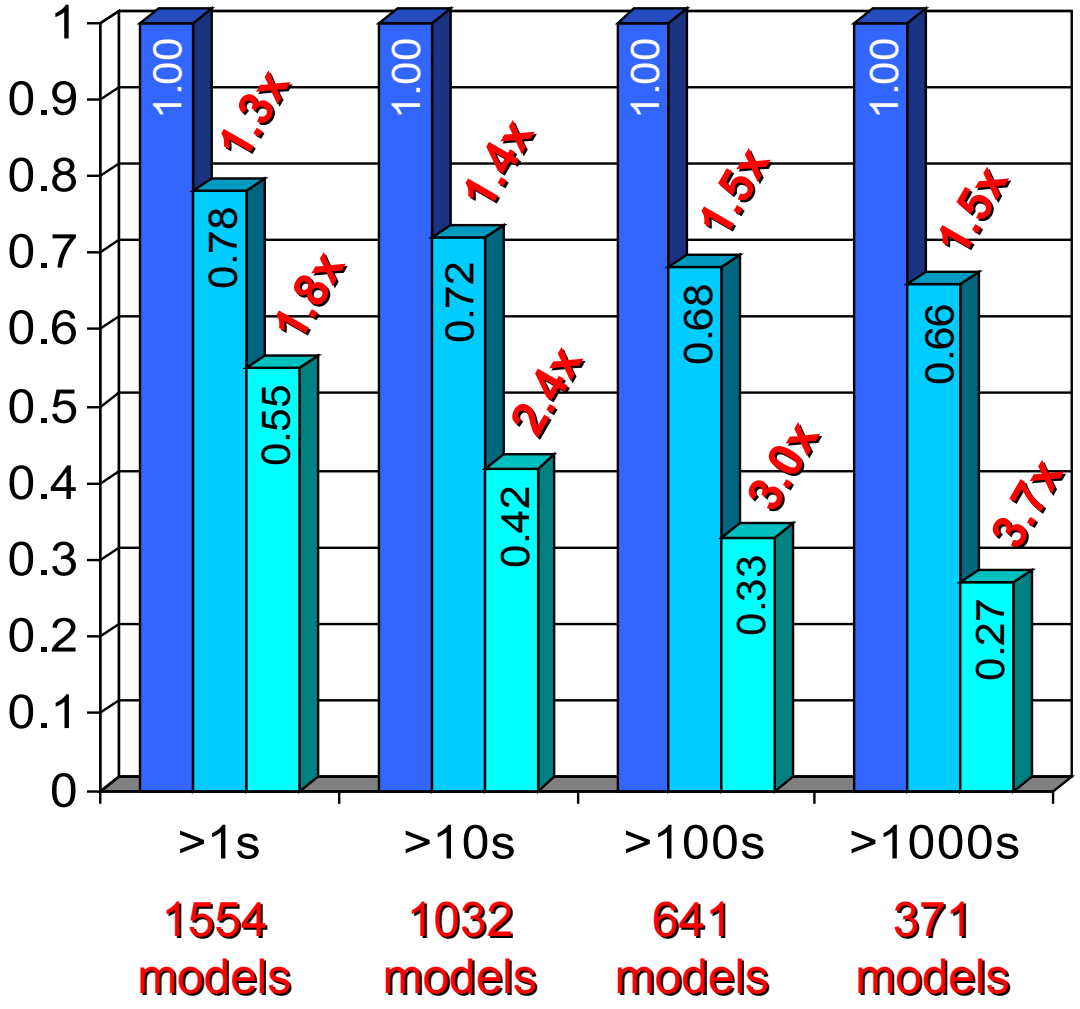
2.70x speed-up

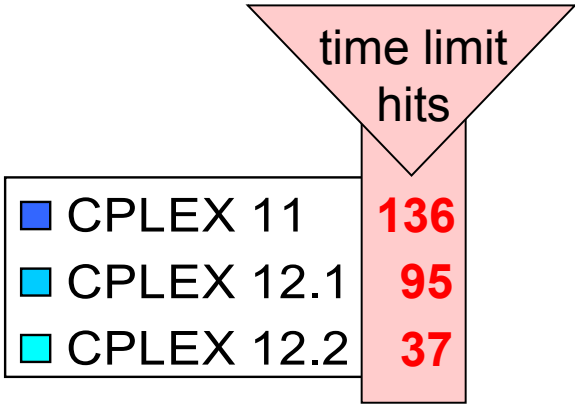
CPLEX benchmark – sequential





CPLEX benchmark – deterministic parallel (4 threads)





Questions & Answers

Thank You