https://doi.org/10.1016/j.ejco.2022.100031

Progress in Mathematical Programming Solvers from 2001 to 2020 and back to the 90s

Thorsten Koch

Timo Berthold, Jaap Pederson, Charlie Vaneret

Zuse Institute Berlin / TU Berlin

Let's SCIP it!

Cray T3D

The T3D (Torus, 3-Dimensional) was Cray Research's first attempt at a massively parallel supercomputer architecture. Launched in 1993, it also marked Cray's first use of another company's microprocessor. The T3D consisted of between 32 and 2048 Processing Elements (PEs), each comprising a 150 MHz DEC Alpha 21064 microprocessor and either 16 or 64 MB of DRAM. PEs were grouped in pairs, or nodes, which incorporated a 6-way processor interconnect switch. These switches had a peak bandwidth of 300 MB/second in each direction and were connected to form a three-dimensional torus network topology.

The hardware landscape two decades ago

2000 the first camera phone (0.11 megapixel). USB flash drives were introduced.

2001 Windows XP is released

the latest CPUs were:

Intel 32-bit Pentium-III ~1 GHz, and Pentium-4 at 1.5 GHz.

IBM 64-bit POWER-7 >3 GHz.

https://commons.wikimedia.org/w/index.php?curid=3961503

2002 Earth Simulator is the worlds fastest Supercomputer with 35,86 Teraflops.

(5120 CPUs, 10 TB RAM, 700 TB storage.)

Today a single DGX-A100 (~$200k) has twice the Teraflops performance.

And getting a system with 700 TB and 10 TB RAM altogether will be below ($500k)

Cost of the ES project was ~$570 Mill. * 1.46 (inflation) => now ~2000 times cheaper.

2003 First PC to allow more than 4 GB memory.

SUN UltraSparc 10

440-MHz UltraSPARC CPU, 64 Bit, RAM 512 MB (max 1GB)

Project Cray to develop a distributed B&C (MIP) Framework for the T3D (in C++)

On the Impact of Communication Latencies on Distributed Sparse LU Factorization.
Martin Grammel, Hans-Christian Hege, Roland Wunderling, 1993

Ein paralleles C++/C-Programmiermodell für ein objektorientiertes B&C-Framework
Martin Grammel, Godmar Back, Roland Wunderling, Hans-Christian Hege, 1994

Paralleler und Objektorientierter Simplex, Wunderling, Roland, 1996

Integer Programs with Block Structure, Alex Martin, 1999 (Parallel SIP using CPLEX LP)

Observations

- Very little progress on OS LP solvers. HIGHS first to challenge CLP
- Really also very little on the IPM front. IPOPT for NLP
- Original author problem prevalent. Managed for SCIP
- Maintenance and Integration question unsolved

The computational landscape has changed and is changing

- Higher clock speed and increased memory bandwidth: accelerate old code, even if not recompiled.
- More efficient processing of instructions:
  superscalar processors, out-of-order execution, branch prediction, and instruction speed-up. Code optimized for an older architecture might not perform optimal on a new one. Recompilation is required.
- New instructions: e.g., FMA, and AVX. To exploit codes needs to be **at least** recompiled.
  highly optimized subroutines, e.g., ATLAS, OpenBLAS, or IMKL, can provide further improvements.
- **Parallel cores and simultaneous multi-threading (SMT):**
  increased max computational performance of a single CPU drastically, but to exploit redesign of the algorithms is needed. There is no automatic benefit for existing non-parallel codes.
- Change from 32-bit to 64-bit addressing/processing:
  allows use of more than 4 GB RAM. No benefit for existing 32-bit codes.
- Improved optimizing compilers:
  from gcc version 2.95 in 2001 to gcc version 10 in 2020 compilers have improved a lot and generate better performing code. Recompilation would be needed to benefit from it.
- Interior Point Methods benefit much more than Simplex
- **GPU** and *Quantum Computing*?

How can we measure progress in Linear and Mixed Integer Programming solvers?

Easy (naïve?) answer:

We measure how long it takes to compute an optimal solution and compare

LP Progress: 1988 – Present

(Operations Research, Jan 20[illegible]—15)

- Algorithms (*m[illegible]ndependent*):

  Primal *vers[illegible]* Primal/Dual/Barrier **3300x**

- Mach[illegible]orkstations →PCs): **1600x**

- **Algorithm × Machine** **5 300 000x**

  (2 months/5300000 ~= 1 second)

How long do we want to wait?

Mathematical Programming 68 (1995) 213–237

Computational experience with a difficult mixed-integer multicommodity flow problem☆

D. Bienstock*, O. Günlük

Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA

Received 27 April 1993; revised manuscript received 17 June 1994

ICONG(2)

$$\min z \qquad k, i, j \in \{1, \ldots, 24\}$$

$$\text{s.t.} \quad \sum_{j \neq i} x_{ij} = 2 \quad \text{for all } i \quad (1)$$

$$\sum_{j \neq i} x_{ji} = 2 \quad \text{for all } i \quad (2)$$

$$f_{kij} \leq M \cdot x_{ij} \quad \text{for all } k, i \neq j \quad (3)$$

$$\sum_{j \neq i} f_{kij} - \sum_{j \neq i} f_{kij} = s_i^k \quad \text{for all i}, k \quad (4)$$

$$\sum_{k} f_{kij} \leq z \quad \text{for all } i \neq j \quad (5)$$

$$f_{kij} \geq 0 \quad \text{for all } k, i \neq j \quad (6)$$

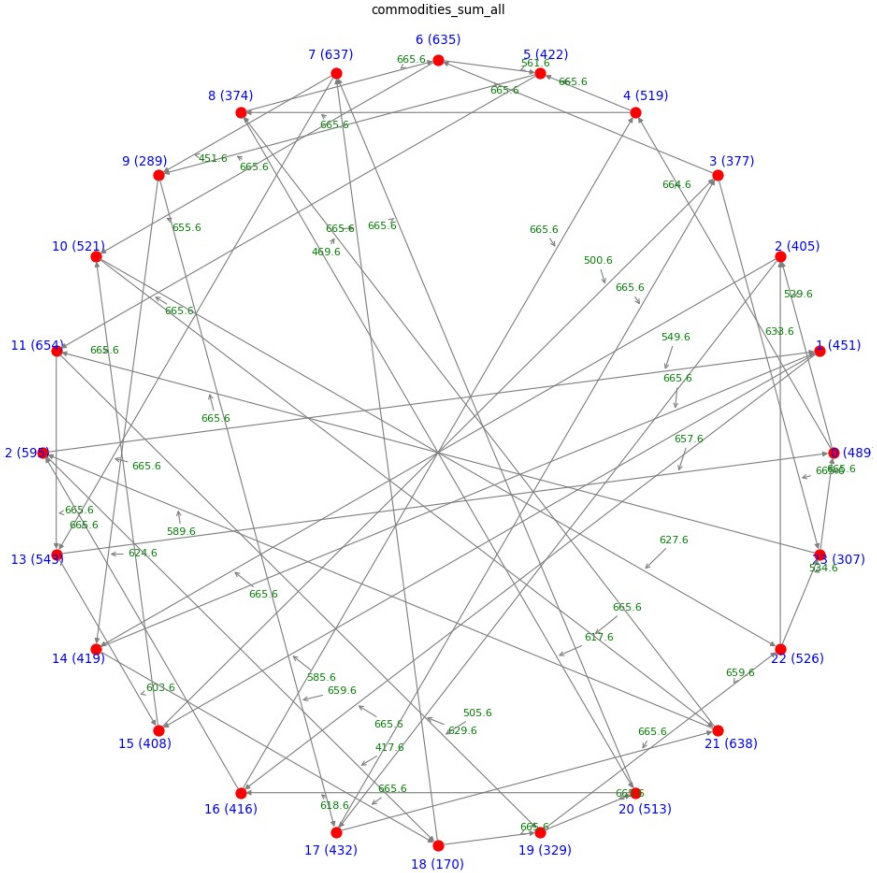
$$x_{ij} \in \{0,1\} \quad \text{for all } i \neq j \quad (7)$$

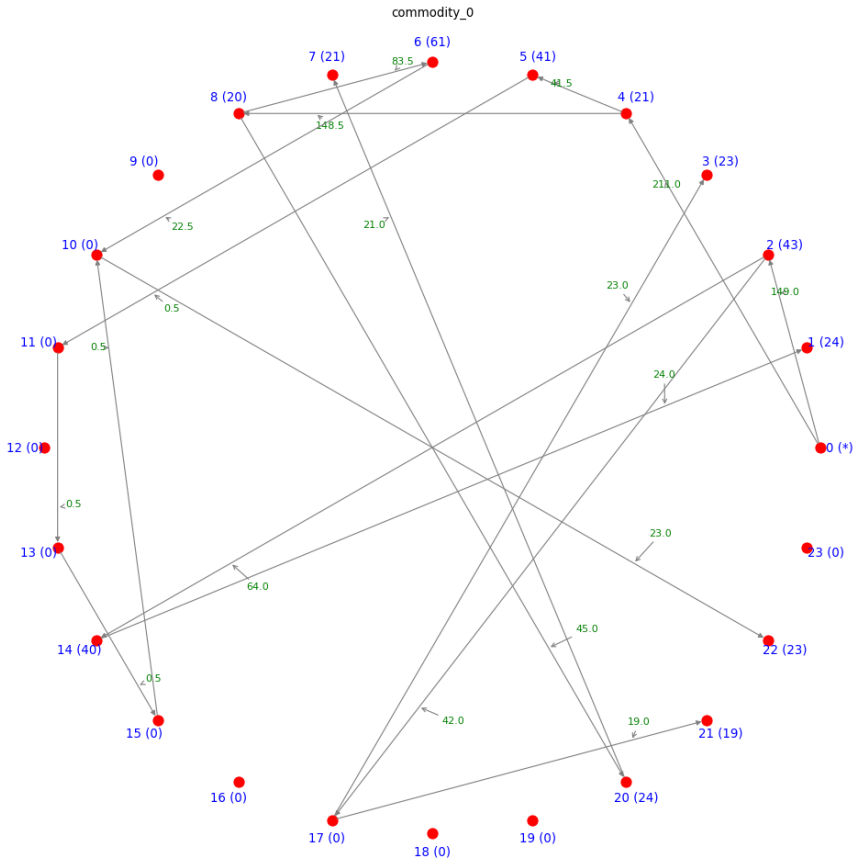
```
Root relaxation objective:  30.28571
dano3mip root relaxation : 576.2316

Best known solution:       665.5714
```

	30.28571	95.4%
	350.63706	47.3%
	425.04748	36.1%
	436.46971	34.4%
Gurobi 9.5.1 root cuts	439.96568	33.9%
original model	577.82468	–
with some cuts removed	577.88492	–

Best know solution





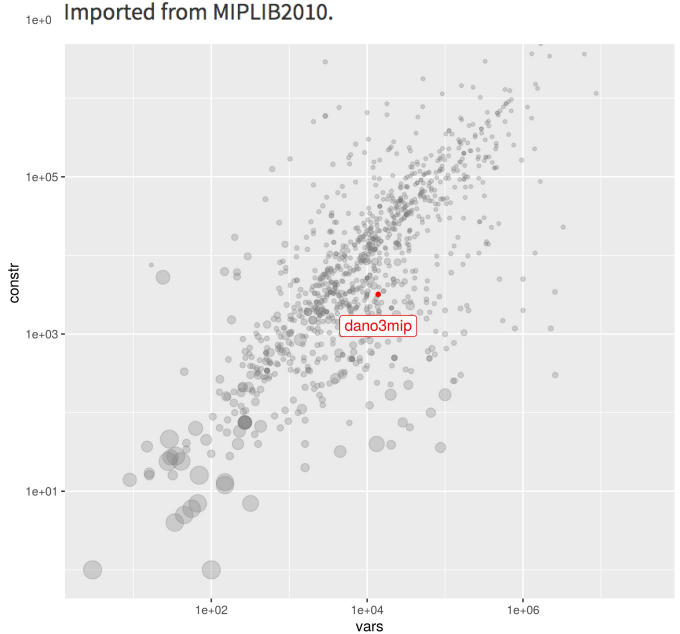
dano3mip

variable_bound | cardinality | mixed_binary

Submitter	Variables	Constraints	Density	Status	Group	Objective	MPS File
Daniel Bienstock	13873	3202	1.79317e-03	open	dano	665.571428571428*	dano3mip.mps.gz

Telecommunications applications

Imported from MIPLIB2010.



Variables	13873
Constraints	3202
Binaries	552
Integers	0
Continuous	13321
Implicit Integers	0
Fixed Variables	0
Nonzero Density	0.00179317
Nonzeroes	79655

MIPLIB 3.0

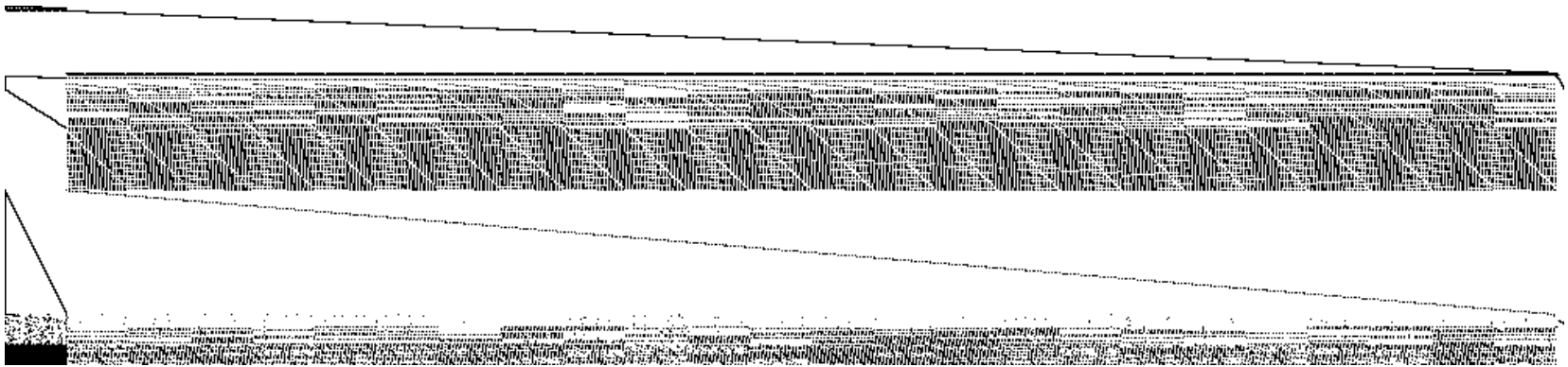
January 1996

: (To download in Netscape, click while pressing the SHIFT key)

- The MIPLIB 3.0 Problem Set as a compressed tar file
- MIPLIB Problem Set:
  - 10teams
  - air03
  - air04
  - air05
  - arki001
  - bell3a
  - bell5
  - blend2
  - cap6000
  - dano3mip
  - danoint
  - dcmulti

Best solutions so far

	ID	Objective	Exact	Int. Viol	Cons. Viol	Obj. Viol	Submitter	Date	Description
3	4	665.5714	665.5714	0	0	0	Yuji Shinano	2020-04-16	Obtained with ParaSCIP in 2014
1	3	667.5577		0	0	0	Edward Rothberg	2019-12-13	Obtained with Gurobi 9.0
4	2	676.5630		0	0	0	Robert Ashford and Alkis Vazacopoulus	2019-12-18	Found using ODH|CPlex
2	1	691.8961	691.8961	0	0	0	-	2018-10-12	Solution found during MIPLIB2017 problem selection.



Some of the “issues”

We observe:

- We have to set a time limit.
- Some instances will not solve to optimality within the time limit on some machine/solver combi.
- Some instances might solve too fast to be measured accurately on some machine/solver combi.
- Measuring CPU time does not work in the parallel case and wall time might have high variability.

Questions that have to be answered:

- Which machine to use?
- Which solver to use?
- Which instances to use?
- Which time limit to use?
- What to do if the time limit is hit?
- Which amount of memory and how many cores?
- But wondersolver can do much better on cursed.lp if you set parameter abracadabra=42
- Can we do the experiment in such a way that we can explain the result?

Our answers

- **Which machine to use?** (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
  Notes:
  - Hyper-Boost and Simultaneous-Multi-Threading leading to high variability.
  - CPU time vs. Wall-clock-time in the parallel case.

Unchanged Network Simplex code: different compile / hardware / load

		Pentium-III 870 MHz	i7-9700 3.6 GHz	empty	full
Name	Iterations	CPLEX 7.0 [s]	CPLEX 7.0 [s]	CPLEX 12.10 [s]	CPLEX 12.10 [s]
16_n14	1,082,628	1369.38	47.70	49.61	81.52
i_n13	2,038,669	954.99	45.30	46.82	76.93
lo10	1,403,624	1086.99	65	64.84	103.32
long15	870,757	1061.19		65.72	93.88
netlarge1	15,443,416	2648.15		115.98	323.30
netlarge2	1,031,653	670.7		42.90	65.17
netlarge3	2,090,956			80.08	134.66
netlarge6	643,226		8.52	7.90	23.05
road_flow_05_WI_a	1,823,212		24.78	24.56	72.31
road_flow_05_WI_b	1,202,7		13.52	13.62	39.81
road_flow_05_WI_c	1	.15	19.36	18.47	57.09
road_flow_07_TX_a		4369.91	194.92	194.00	630.08
road_flow_07_TX_b	6,	4702.96	207.94	206.38	673.12
road_flow_07_TX_c	6,5	4570.84	206.14	205.96	655.00
square15	918,010	1192.27	72.60	71.56	111.16
wide15	870,757	1059.15	62.30	66.92	99.60
Shifted geom mean (1)		**1145.92**	**55.56**	**55.83**	**121.10**

Conclusion:
Hardware speed-up 20 x

Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- **Which solver to use?** Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best you can do at a given moment in time.
  Note: Not focusing on a particular solver makes it much harder to explain in detail what happened.

Our Solver Ensembles

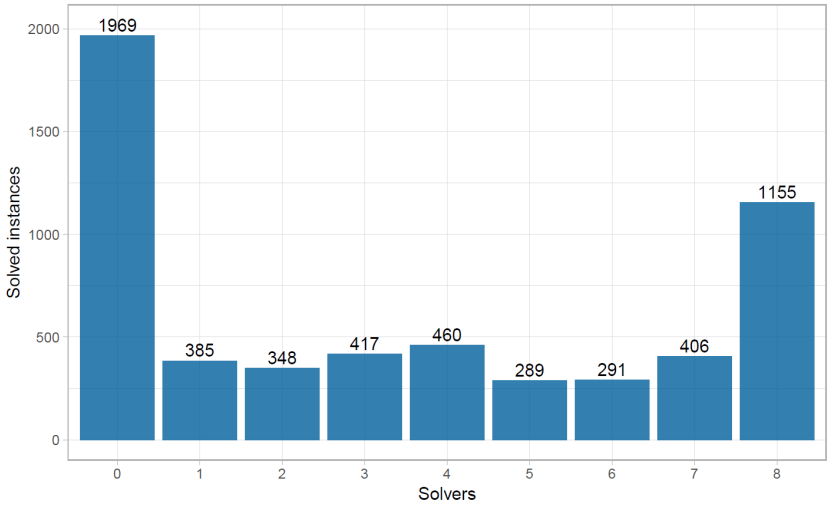
OLD is the best result we could get out of

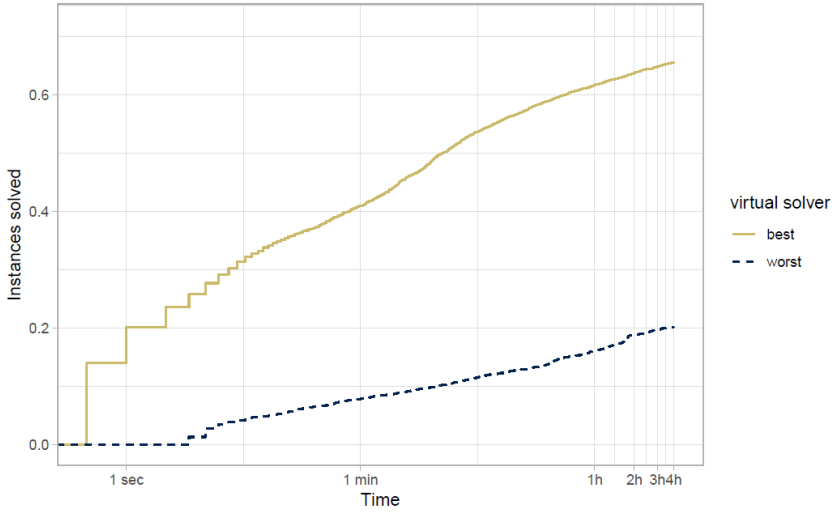
- CPLEX Linear Optimizer 7.0.0 (2000),
- Xpress-MP Hyper Integer Barrier Optimizer Release 14.10 (2002), and
- MOSEK Version 3.2.1.8 (2003).
- Codes use parallel computations only for IPM LP solvers.

NEW is the best of

- IBM ILOG CPLEX Interactive Optimizer 12.10.0.0 (2019),
- Gurobi Optimizer version 9.1.0 (2020),
- FICO Xpress Solver 64bit v8.11.0 (2020), and
- COPT Version 1.4.7 (2020). The later was only used for LPs
- All run with both a single and eight threads.

Experience from MIPLIB 2017





- Most instances can either be solved by all solvers or by none
- Interestingly, even spread on buckets 1-7
- Instance complexity increases logarithmically
- Virtual best solves more instances in 2 seconds than virtual worst in 4 hours!

Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- **Which instances to use?**
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3

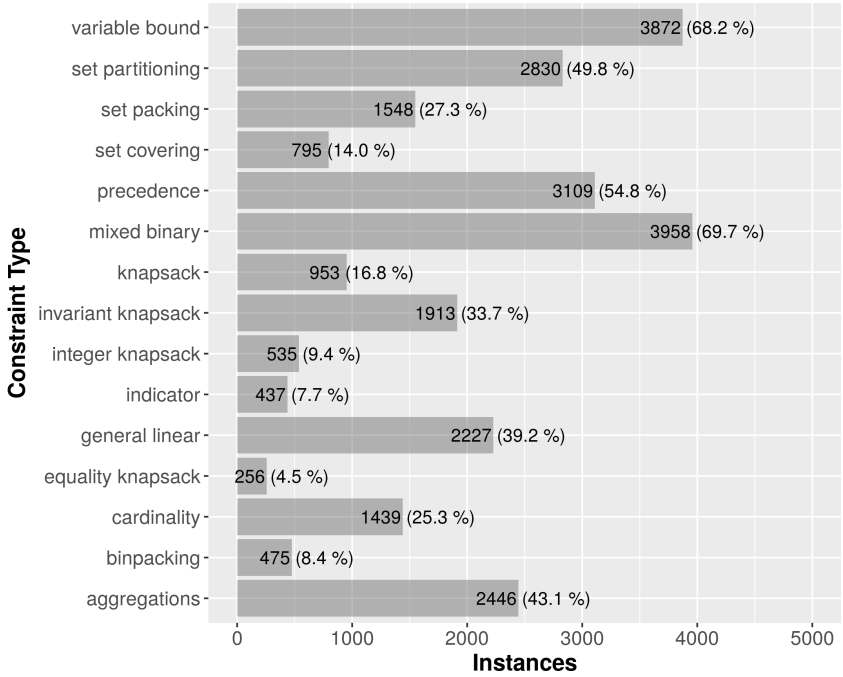
Choosing instances

	B&B nodes		Time [s]		
Name	**OLD/P-III**	**New/i7**	**OLD/P-III**	**New/i7**	**Speed-up**
nw03	131	24	54	1.62	33
mas76	467,454	192,719	198	3.50	57
neos-1122047	64	1	79	1.28	62
mod011	16,671	3,288	926	5.77	160
air05	1,961	2,189	440	2.11	209
qiu	36,452	4,434	1,393	1.49	935
cap6000	16,768	1	268	0.10	2,680
bell5	687,056	915	411	0.93	13,700
neos1171737	28,354	1	116,745	2.67	43,725

MIPLIB 2017: Two sets

Collection Set

- ▷ good coverage of submissions
- ▷ **balanced representation of instance features**
- ▷ including numerically difficult instances



Benchmark Set (which we used)

- ▷ subset of collection
- ▷ excluding numerically difficult instances
- ▷ limit size for running time constraint
- ▷ **balanced representation of MIP solver performance**

An instance is benchmark-suitable, if:

- it can be solved by at least 1 considered solver within 4 hours (not too hard)
- it requires at least 10 seconds for at least 50% of the solvers (not too easy)
- it has at most 10^6 nonzero entries (not too big)
- it has a constraint and objective dynamism of at most 10^6 (not too different scales)
- it has only absolute matrix coefficients smaller than 10^{10} (stable coefficients)
- the solution (objective) value is smaller than 10^{10} (reasonable objective scale)
- the results of all solvers are consistent (numerically stable)
- it has no indicator or SOS constraints (“pure” MIP)
- it is bounded (otherwise pure presolving/LP problem)

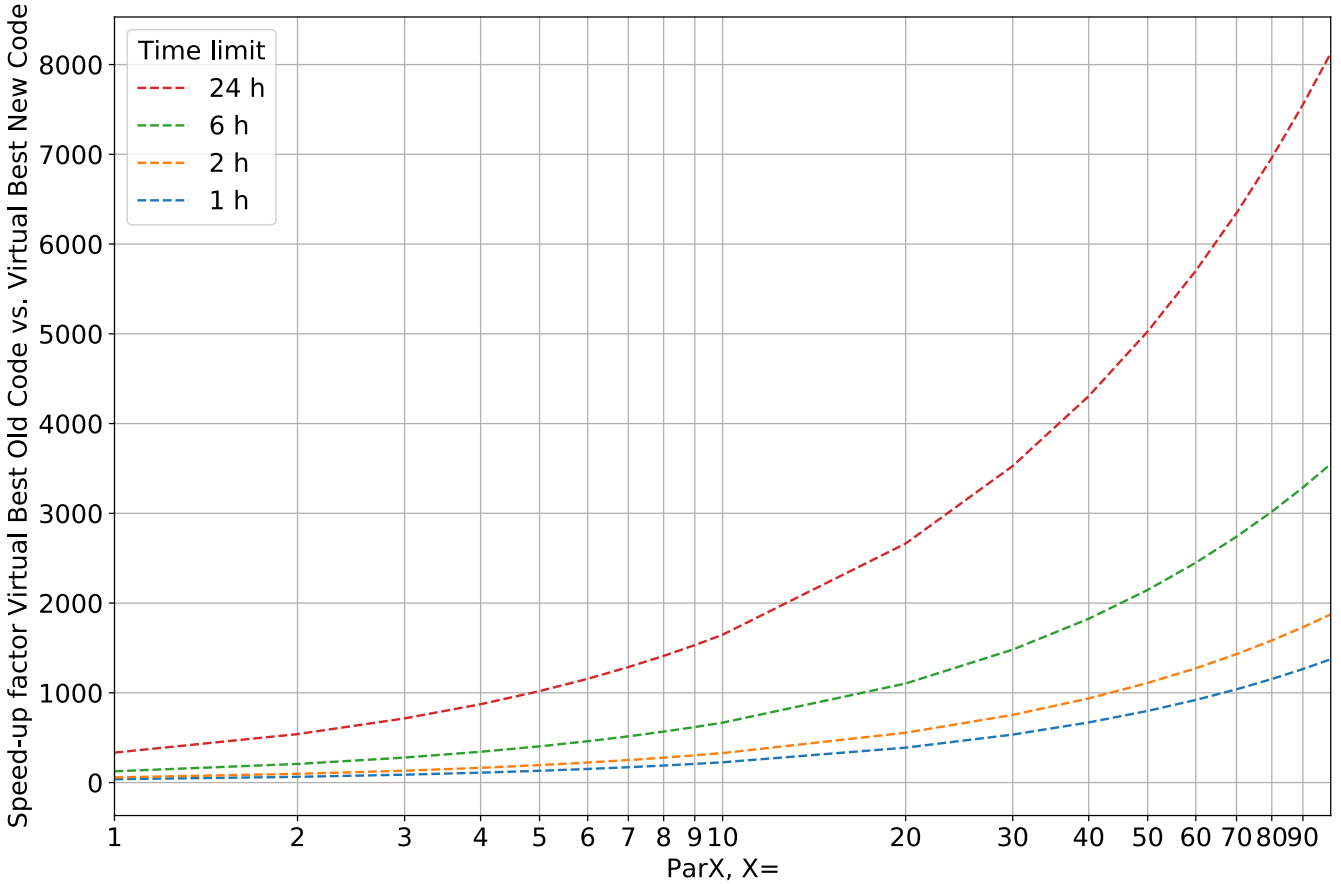
Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- Which instances to use? Difficult
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3
- Which time limit to use? Where is the quiescence point? Take speed-up ratio into account.
  We started with two hours and then extended basically to one day.
  It was helpful that OLD is basically never faster than NEW.

Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- Which instances to use? Difficult
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3
- Which time limit to use? Where is the quiescence point? Take speed-up ratio into account.
- What to do if the time limit is hit?

How to handle time limits?



Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- Which instances to use? Difficult
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3
- Which time limit to use? Where is the quiescence point? Take speed-up ratio into account. Min time.
- What to do if the time limit is hit? PAR1

Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- Which instances to use? Difficult
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3
- Which time limit to use? Where is the quiescence point? Take speed-up ratio into account. Min time.
- What to do if the time limit is hit? **PAR1**
- Which amount of memory and how many cores? Old has only 4 GB max anyway.

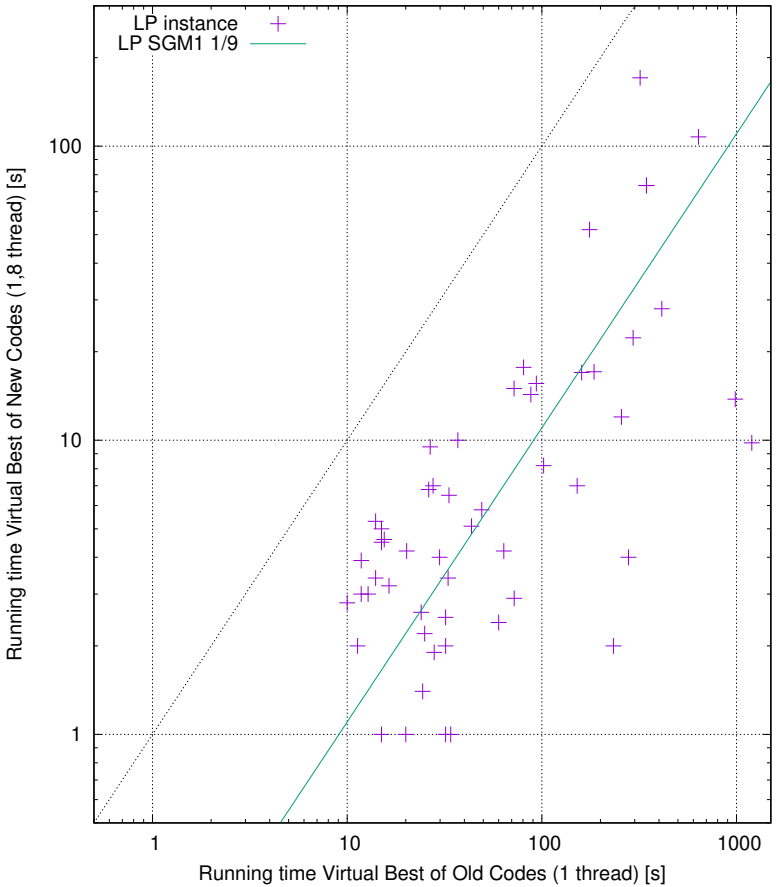
Our answers

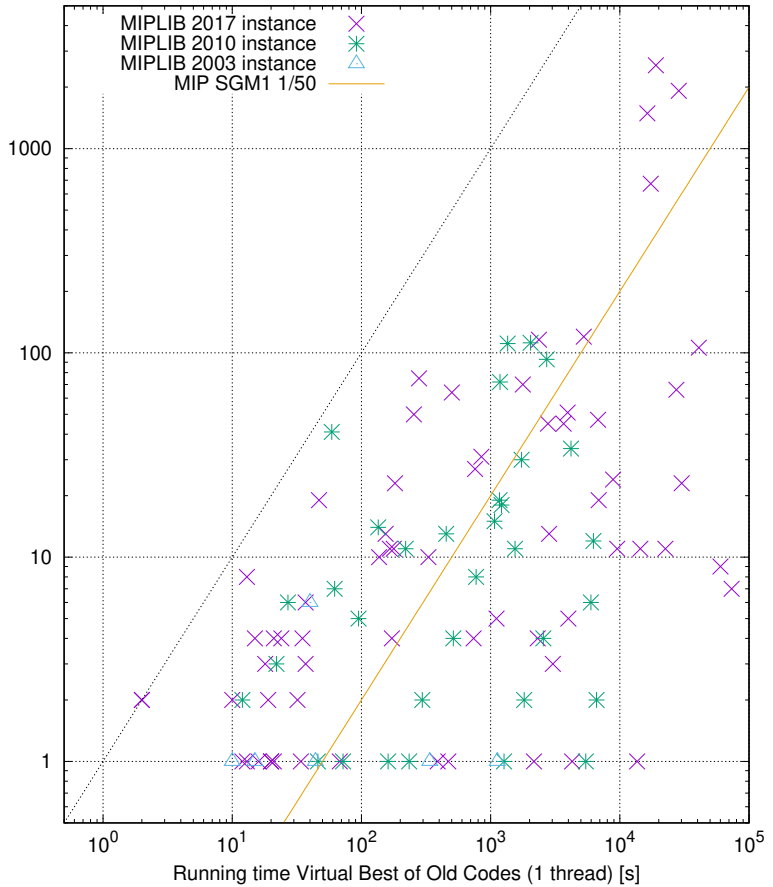
- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- Which instances to use? Difficult
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3
- Which time limit to use? Where is the quiescence point? Take speed-up ratio into account. Min time.
- What to do if the time limit is hit? PAR1
- Which amount of memory and how many cores? Old has only 4 GB max anyway.
- But wondersolver can do much better on cursed.lp if you set parameter abracadabra=42.
  Impossible to do fair, unless you do this like, e.g., SPEC, where the solver vendors submit the parameter settings. Also, not so useful in an ensemble approach.

Our answers

- Which machine to use? (What we could find)
  Pentium-III 870 MHz and 8-core, 8-thread Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB of RAM
- Which solver to use? Two choices: Either measure the progress of a particular solver, or as we did, measure progress of the virtual best available solver (ensemble approach),
  since this is the best, you can do at a given moment in time.
- Which instances to use?
  See *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Gleixner et al., 2021 , doi 10.1007/s12532-020-00194-3
- Which time limit to use? Where is the quiescence point? Take speed-up ratio into account. Min time.
- What to do if the time limit is hit? PAR1
- Which amount of memory and how many cores? Old has only 4 GB max anyway.
- But wondersolver can do much better on cursed.lp if you set parameter abracadabra=42
- Can we do the experiment in such a way that we can explain the result?
  Split hardware and software speed-up.
  Ensemble approach is not making this easier.

Algorithmic progress





Real progress: MIPLIB 2017 Instances that cannot be solved by the old solvers

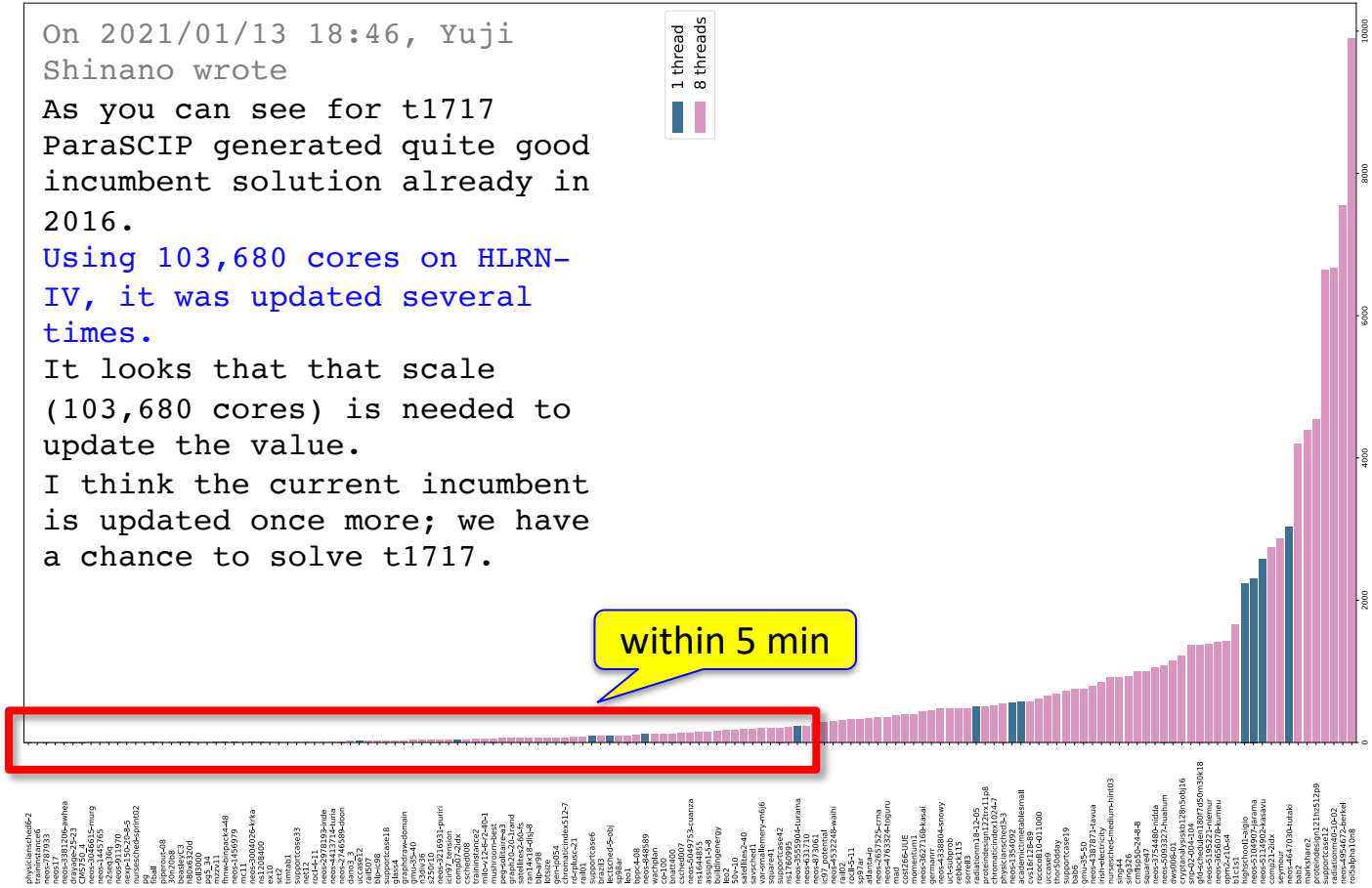
On 2021/01/13 18:46, Yuji Shinano wrote

As you can see for t1717 ParaSCIP generated quite good incumbent solution already in 2016.

Using 103,680 cores on HLRN-IV, it was updated several times.

It looks that that scale (103,680 cores) is needed to update the value.

I think the current incumbent is updated once more; we have a chance to solve t1717.



There are a still instances from MIPLIB 2003 that could not be solved in the old times and can‘t be solved now:

- dano3mip
- liu
- momentum3
- protfold
- t1717

- ds (by HPC)
- mkc (by reformulation)
- stp3d (by HPC)
- swath (hard)
- timtab2 (by special)

1. Solvability

- The likelihood that a MILP instance solves at all (out-of-the-box) is vastly increased.
- 156/240 instances (62%) that could be solved by NEW, could not be solved by OLD at all.

2. Speed

- Hardware got ~20 times faster. Solvers ~50 time faster. LP ~9 times.
  Altogether about 180/1000 times! This is ~42% per year.
- What took long in 2001 speed-up more on average.
- Whatever took 3 min (LP), 15 min (MILP) in 2001 is now done in just 1 second.
- What took a day is now down to 1.5 min.

3. Size

- When there was 1 GB in 2001, there is now 1 TB. We can solve much larger LP instances.

Thank you very much

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

MODAL
FORSCHUNGSCAMPUS

RESEARCH
CAMPUS
Public-Private Partnership
for Innovation