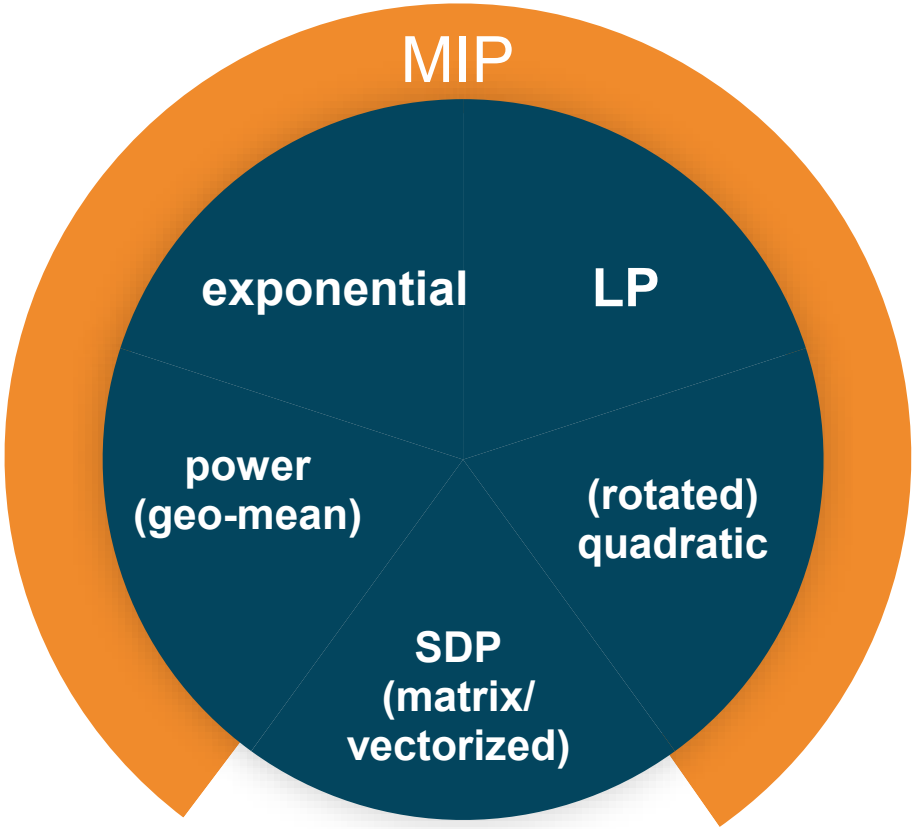
What's new in

Jens Schulz
COO – Mosek ApS

GOR 2026, Passau, Sep 1-4, 2026

Inside MOSEK

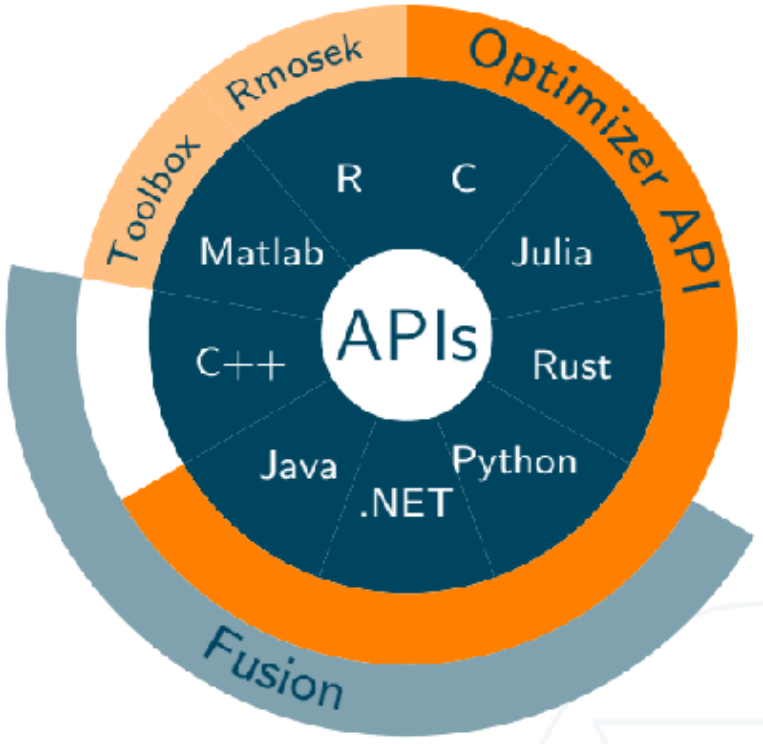
- The company was established in 1997, releasing the first version of the solver two years later, in 1999.
- Simplex / Interior-point algorithms for continuous problems.
- (Conic) Branch-and-Cut / Outer-approximation algorithms for discrete problems.



mosek

Available via various open-source and commercial APIs

- Matrix-oriented Optimizer APIs
- **Fusion API**: Thin and efficient object-oriented API layer resembling math formulation for large-scale **conic models**



- **Third-Party:** https://www.mosek.com/resources/third-party/
  - Commercial: **AMPL, GAMS**
  - CVXPY, CVXR, Pyomo, PuLP, PyOpt, linopy/PyPSA, Julia/JuMP, …

mosek

1997: Mosek ApS was founded by Erling and Knud Andersen

1999: First release (LP, QCQP, SOCP & parallelization build-in)

2000: Extended APIs: Java, .NET

2002: First full-time employee working on Simplex Optimizer

2007: First Python interface

2012: Mosek modelling manual

2013: SDP, MIP, Fusion API

2016: OptServer

2019: exponential and power cones

2024: Pythonic Fusion API, Julia, Rust, OptServerLight,
disjunctive constraints, generalized power cones, geometric mean cones

2026/7: *coming soon:* FreeSimplex, Folding, Pre-root heuristics

mosek

What's new in MOSEK 12 – preliminary; beta release in Autumn 2026

Linear & conic optimizer

- Modernized Primal and Dual Simplex Optimizers
  - "Free Simplex" runs both in parallel
  - Supports emulated 128bit floating precision mode for ill-conditioned basis
- Modernized code of interior point optimizer:
  - Exploit AVX instructions on AMD hardware
  - More robust and faster in basis identification esp. for large problems
- **LP folding** broadened to interior-point optimizer with basis identification and Simplex

Mixed-integer optimizer

- Vastly improved conflict analysis and better conflict handling
- Better propagation algorithms for symmetry handling
  - Detection of orbitopes (special cases)
  - Improved space and time efficiency
- Pre-root (LP-independent) heuristics, inspired by feasibility jump and shift-and-propagate paradigm
- Improved detection of implied integers

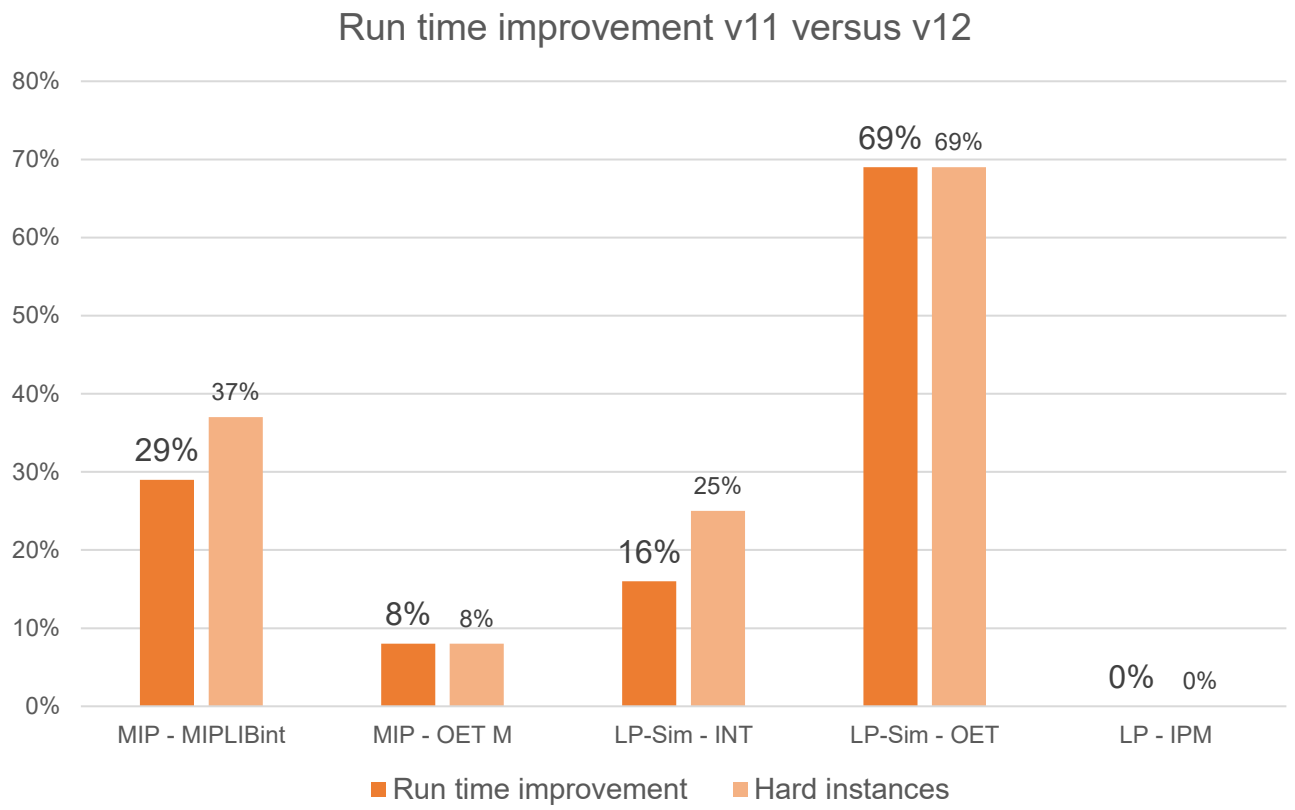
mosek

Our benchmark sets

- Goal of benchmarking:
  - Identify impact of features
  - Numerical stability and performance improvements for customers
  - Not targeted to prove “better than X”
  - Avoid overtuning -> sample from MIPLIB and others
- Therefore, benchmark sets for LP, MIP, MIP heuristics,… differ
- MIPLIB2017: standard from H. Mittelmann; removed too easy and too hard ones (re time limit)
  - MIPLIBlp: solve root node LP on selection of ~1,200 instances
  - MIPLIBint: internal selection of 240 instances with 3 seeds -> 720 instances
- LP benchmarks for Simplex:
  - 2,000+ instances from MIPLIB2017, NetLib & few others (customers) – only solve root
- OET: OpenEnergy Transition Project https://openenergybenchmark.org/
  - We focus on realistic problems for LP and MIP; Sets clustered into S, M & L

mosek

LP & MIP benchmarks release over release – v12 Beta



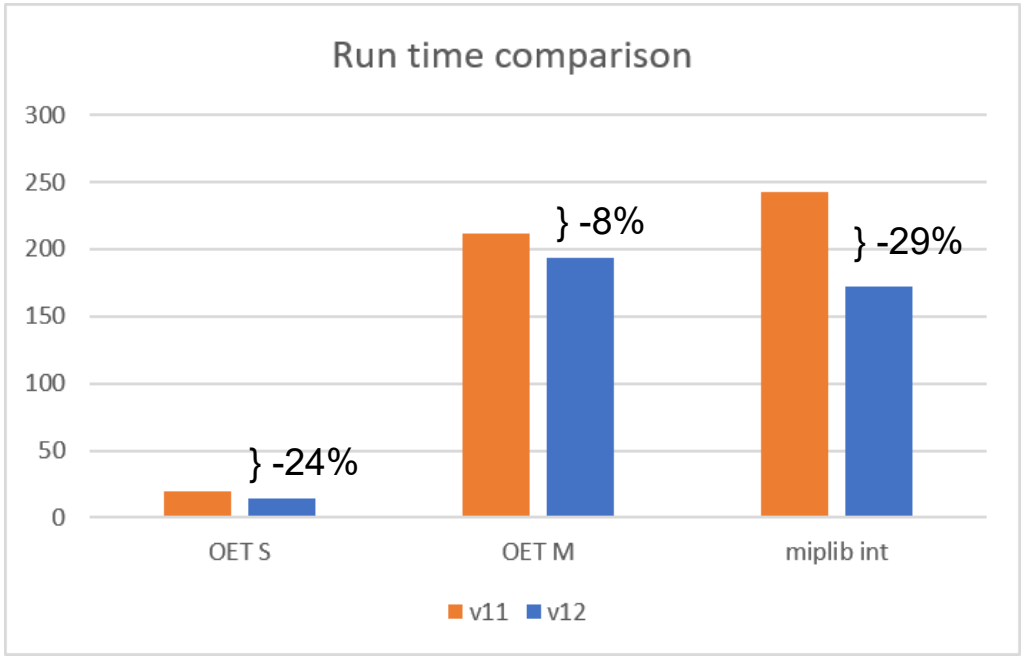
Runtime improvement measured in GMST (shifted geometric mean) for MIP and LP on respective test sets
In light orange, natural subsets with different performance, e.g., hard instances or long-running instances

mosek

MIP – Overall

Benchmark results for MIP – release over release improvement

- All tests run on
  - Intel(R) Xeon(R) E-2124 CPU
  - 32 GB ram & 4 threads
  - max time 2h; relative gap limit 1e-9
  - Each instance is run with 3 random seeds
  - Shifted geometric mean with offset s = 1sec
  - MIPLIBint: 720 instances from MIPLIB2017 (excl too easy, too hard,.. and run with 3 seeds)

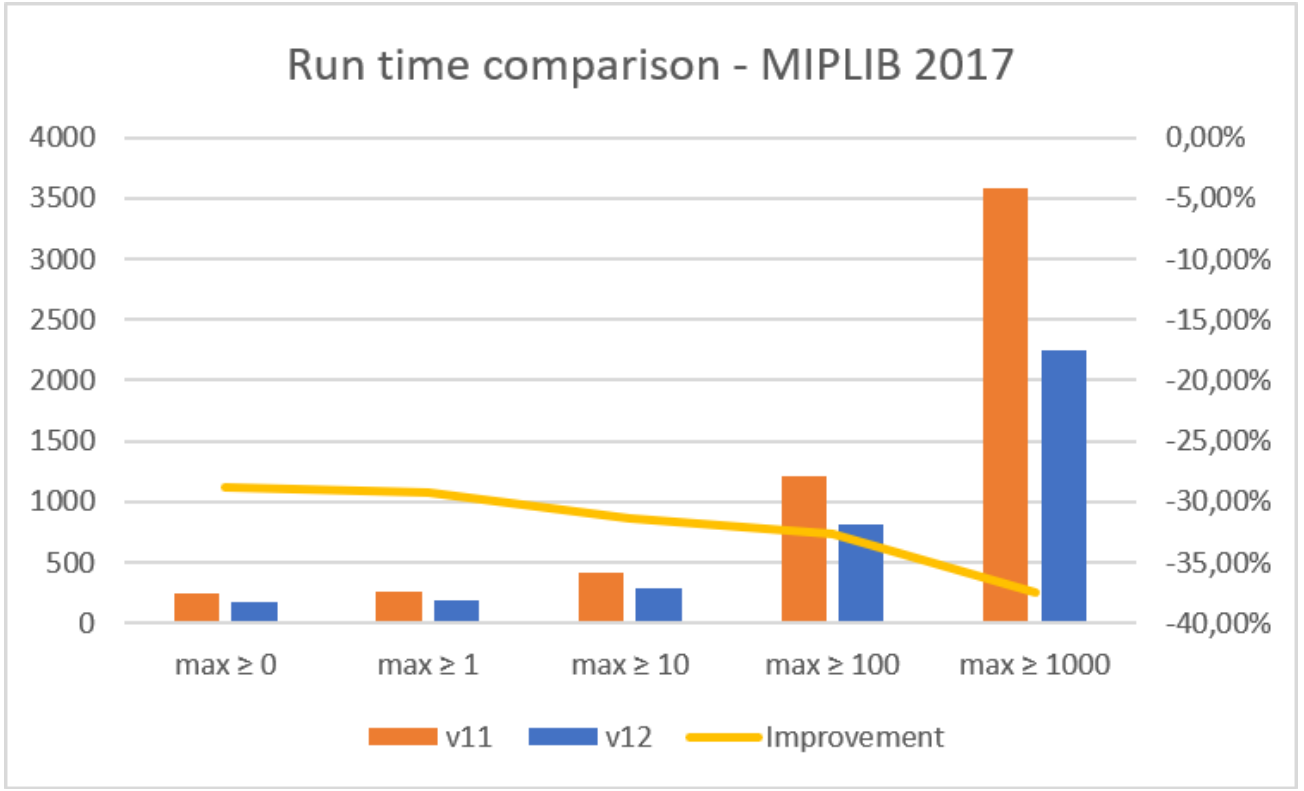


		Solved		Run time GMST			WINS (5%)	
Instance set	#instances	v11	v12	v11	v12	Improvemen	v11	v12
OET S	78	78	78	19,49	14,73	-24,42%	30	47
OET M	102	98	93	211,35	193,90	-8,25%	37	57
miplib int	441	392	417	242,34	172,62	-28,77%	129	281

mosek

Run time comparison of all versus long-running

- Detailed run time comparison demonstrates significant gains for harder instances, the longer the better v12 performs
  - MIPLIBint
    From 28.77% -> **37.5%** improvement for instances that take at least 1,000 sec for one solver



mosek

MIP – pre-root

Results for MIP Pre-Root Heuristics

- New pre-root (LP-independent) heuristics: inspired by feasibility jump and shift-and-propagate paradigm
- Take-aways:
  - Good for finding good solutions early, and any solution at all
  - Limited impact on overall solving time
  - Impacted instances from miplib2017:
    - of combinatorial nature where LP solution is no good starting point
    - edge-coloring on graphs, timetabling, (satellite) scheduling, network problems, ...
  - Energy instances less affected

- # feasible solutions **after root**:

		# feas sol after root		
	instances	w/o	with	Improvement
MIPLIB2017	720	482	516	7%
MIPcc26	106	62	77	24%

- **Time to first feasible solution** heavily reduced by above 80-90%

	instances	w/o	with	Improvement
MIPLIB2017	652	5,84	3,77	35%
MIPLIB2017 aff	178	5,58	0,75	87%
MIPcc26	93	44,76	9,93	78%
MIPcc26 aff	41	146,62	6,56	96%

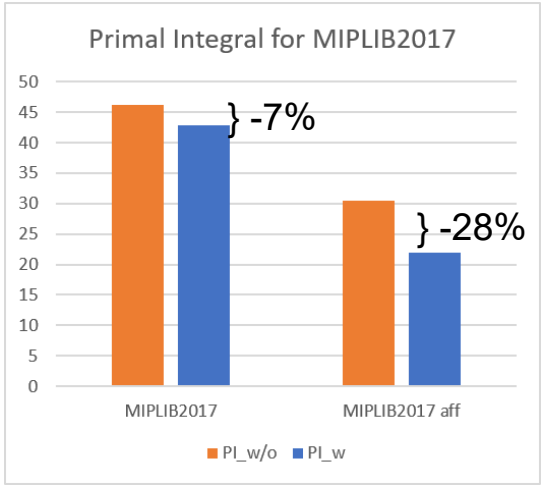
mosek

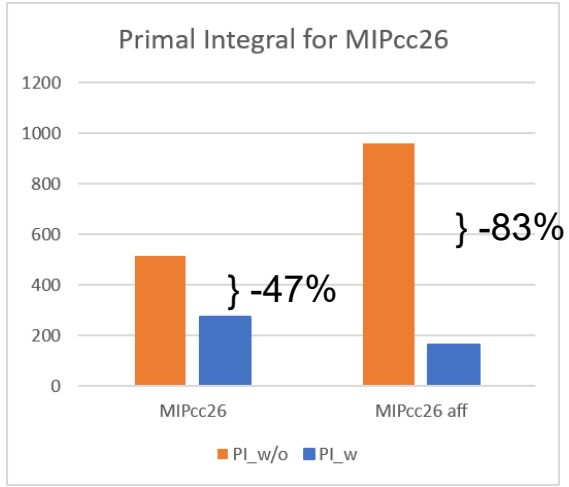
MIP – Pre-root heuristics

- Primal integral P(t_max):
  - T. Berthold, ISMP 2012
  - favors finding good solutions early
  - considers each update of incumbent
  - Notion of “average solution quality” (P(t_max)/t_max)
  - expected quality of the incumbent, if stopped arbitrarily
- Primal integral is the integral over the primal gap function that monotonically decreases when a new incumbent is found.

- **Primal integral** reduced by up to 83% on MIP competition data set

	instances	PI_w/o	PI_w	Improvement
MIPLIB2017	714	46,2	42,83	7%
MIPLIB2017 aff	173	30,49	21,91	28%
MIPcc26	103	516	273,53	47%
MIPcc26 aff	37	961,8	163,61	83%





mosek

Folding

Impact of folding

- Idea: Use equitable partition by defining disjoint row and column sets, and solve aggregated problem
- To obtain a basic solution, a potentially expensive basis identification (~crossover) step may be required
- Idea in Mosek: Apply folding in presolve if deemed worthwhile

- For combinatorial problems (e.g., in MIPLIB) folding is often more applicable than in industrial instances from supply chain, energy, finance and forestry.
- Not predictable in advance
- The requirement of basis identification (=crossover) limits the value of folding

mosek

Results for folding using MIPLIB2017

- Average folding factor is 0.78 (not shown in the table). Quite high!
  -> for 78% of instances from MIPLIB2017 folding applies
- Folding leads to improvements in the total run time (~10%), geometric mean of the shifted time ratios and wins.
  - GMST: Geometric mean of shifted time. (shift=0.01)
  - GMRST: Geometric mean of ratio of shifted time.
  - # wins: Number of wins (a win is within 5% of the best).

Fold	Min	Max	Count	Total time(s)	GMST	GMRST	# wins
No	0.00	4375.74	1052	45417.63	0.65	1.00	764
Yes	0.00	3824.17	1052	40008.32	0.55	0.84	908

mosek

Modernized Simplex

Modernized Simplex

- Main challenges: numerically troublesome instances, and large instances
- New starting point heuristic:
  - Start from a basis that is very sparse, stable, and quick to obtain.
  - Per iteration, we try to stay sparse on the path.
  - Replaces a sophisticated method.

- Experiments reveal:
  - Starts far from target
  - Quickly catches up and takes the lead
  - Returns more sparse and numerically stable solutions (if there are multiple optimal points).

- **Impact:** on small easy instances where old starting point heuristic found solution close to optimum, Simplex now runs a bit slower

mosek

Mosek 12 – improvements for primal and dual Simplex

- Primal Simplex

		GMST ratio		Wins (5%)	
	#	legacy	new	legacy	new
All	2023	1.0	0.57	121	1425
Easy	962	1.0	0.89	63	440
Challenge	1061	1.0	0.38	58	985

Easy max time ≤ 0.1 seconds.

GMST geometric mean of shifted times ($+0.01$ s)

- Dual Simplex

		GMST ratio		Wins (5%)	
	#	legacy	new	legacy	new
All	2105	1.0	0.84	366	1189
Easy	1077	1.0	0.93	105	473
Challenge	1028	1.0	0.75	261	716

Easy max time ≤ 0.1 seconds.

GMST geometric mean of shifted times ($+0.01$ s)

Internal test set: MIPLIB; netlib problems + few others

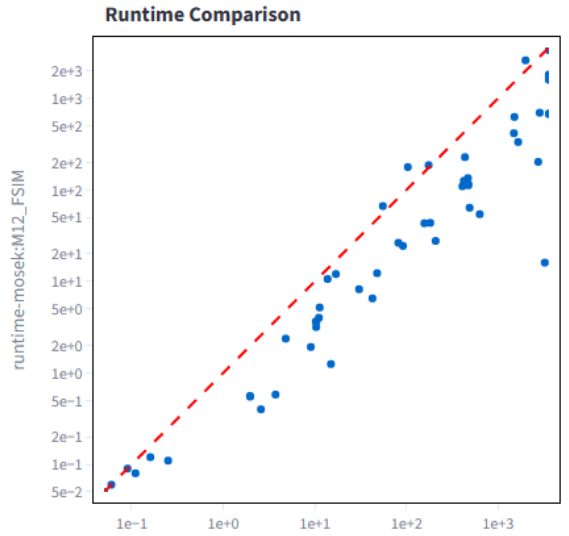
mosek

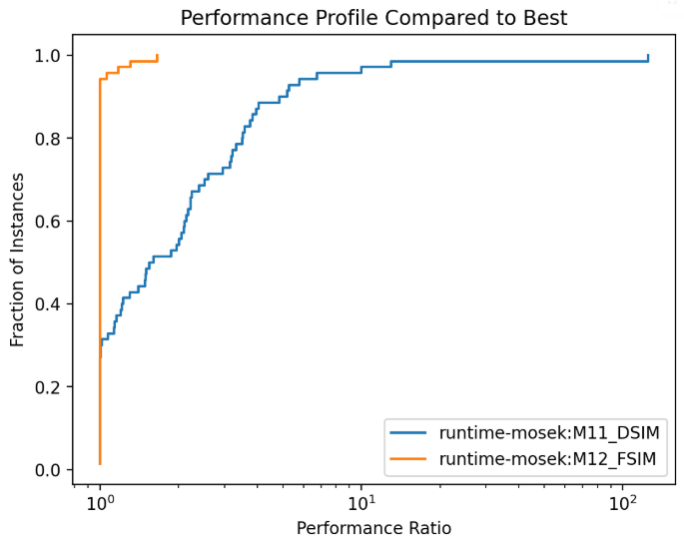
Performance improvement via Free Simplex on energy instances

- In general, dual Simplex in v12 ix slower for very easy problems and faster on harder problems

- Test on internal testset show 50% total runtime reduction in dual Simplex, 66% on primal Simplex
- OET benchmarks S&M: Free Simplex 69% faster compared to v11 dual Simplex

Runs	Min	Max	Count	Total time(s)	GMST	GMRST	# wins
mosek:M11_DSIM	0.05	3243.65	47	21089.23	39.46	1.00	7
mosek:M12_FSIM	0.05	2608.34	47	6638.15	12.28	0.31	43





mosek

Interior point method - updates

Comparison for IPM – machine dependency

- Modernized code of interior-point optimizer:
  - Exploit AVX instructions on AMD hardware
  - More robust and faster in basis identification esp. for large problems

- Fast macOS: (preliminary)
  v12 is ~1% faster, yet has an outlier on MIPLIBint

Runs	Min	Max	Count	Total time(s)	GMST	GMRST	# wins
mosek:112	0.00	428.94	2,256	9717.10	0.13	1.00	1,313
mosek:120	0.00	2506.20	2,256	9741.68	0.12	0.99	1,316

- Linux server:
  v12 is 4% slower in GMST, but faster in total on MIPLIBint

Runs	Min	Max	Count	Total time(s)	GMST	GMRST	# wins
mosek:112	0.00	1276.59	2,253	27099.97	0.32	1.00	1,578
mosek:120	0.00	1234.22	2,253	24065.74	0.34	1.04	1,055

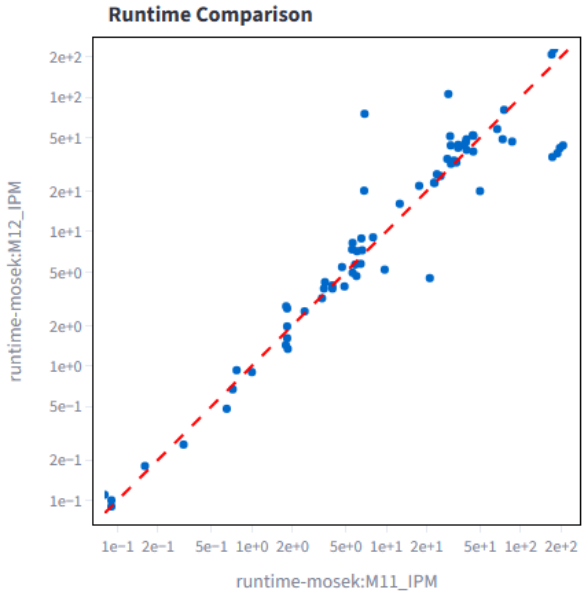
mosek

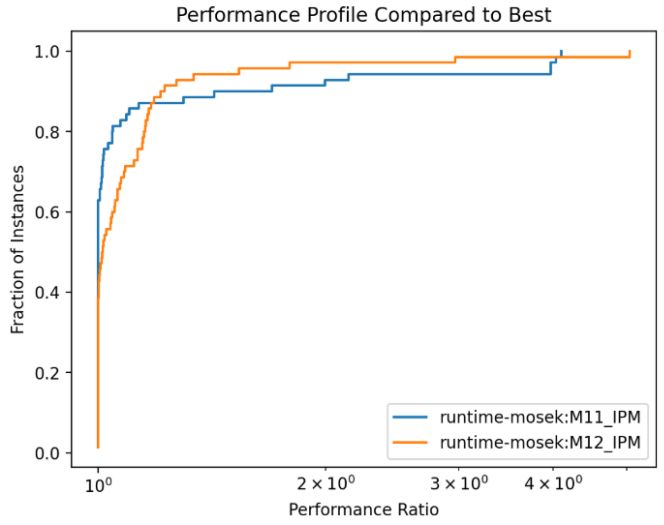
Comparison for IPM – Energy benchmarks

- Modernized code of interior-point optimizer:
  - Exploit AVX instructions on AMD hardware
  - More robust and faster in basis identification esp. for large problems

- LP energy benchmarks (OET - S & M):
  - Stable on average, specific instances benefit from faster basis identification, especially large ones
  - 20% total time reduction, but less in GMST
  - More investigation needed

Runs	Min	Max	Count	Total time(s)	GMST	GMRST	# wins
mosek:M11_IPM	0.08	207.07	70	2313.34	9.60	1.00	47
mosek:M12_IPM	0.09	228.98	70	1952.65	9.47	0.99	33





Other

Provisional release plan

- Early autumn 2026:
  Public beta-version
- Winter 2026:
  full release

mosek

MOSEK

Our solver is the answer for all your LPs, QPs, SOCPs, SDPs, and MIPs. Includes interfaces to C, C++, Java, MATLAB, .NET, Python, Julia, Rust and R.

$$C := \{x \in \mathrm{R}^3 : x_1 \geq \sqrt{x_2^2 + x_3^2}\}$$

Discover | Try | Buy

Academic Licenses:

https://www.mosek.com/products/academic-licenses/

mosek

GOR

PMO111

Optimization in the era of Generative AI

Munich, Germany
November 26 & 27, 2026
Hosted by Flix SE

Source: AI-generated image

PMO112

Shaping the Mobility of the Future: A Mathematical Optimization Perspective

Ingolstadt, Germany
March 11 & 12, 2027

Organized with TH Ingolstadt

Source: https://unsplash.com/de/fotos/luftaufnahmen-von-betonstrassen-7nrsVjvALnA

https://www.gor-ev.de/arbeitsgruppen/praxis-der-mathematischen-optimierung