

INTERACTIVE OPTIMIZATION & LEARNING
[HTTPS://IOL.ZIB.DE](https://iol.zib.de)

Race, Exchange, Improve:

Finding high quality MIP solutions in short time

Gioni Mexi

joint work with Daniel Rehfeldt

Zuse Institute Berlin

OR 2026, Passau · September 1–4, 2026



- I The SCIP Optimization Suite**
and upcoming features in SCIP 11.0
- II Parallel Racing in SCIP and ReXi**
Race · eXchange · improve

The SCIP Optimization Suite

- ◇ One of the fastest open-source solvers for MIP and MINLP
- ◇ Constraint integer programming: Techniques from MIP + CP + SAT + MINLP
- ◇ Basis for specialized solvers: GCG, SCIP-SDP, SCIP-Jack, QuBowl, ReXi, ...
- ◇ ...and much more: Benders decomposition, exact MIP, IIS, reoptimization, concurrent solving, ...



www.scipopt.org

- ◇ One of the fastest open-source solvers for MIP and MINLP
- ◇ Constraint integer programming: Techniques from MIP + CP + SAT + MINLP
- ◇ Basis for specialized solvers: GCG, SCIP-SDP, SCIP-Jack, QuBowl, ReXi, ...
- ◇ ...and much more: Benders decomposition, exact MIP, IIS, reoptimization, concurrent solving, ...
- ◇ A platform for researchers: plugin-based, implement and test your own methods in a general-purpose solver
- ◇ Open source, Apache 2.0



www.scipopt.org

SCIP is a plugin-based

constraint handlers	34	primal heuristics	67	file readers	28
separators	22	presolvers	19	branching rules	16
expression handlers	13	propagators	12	node selectors	8
event handlers	6	cut selectors	4	relaxators	2

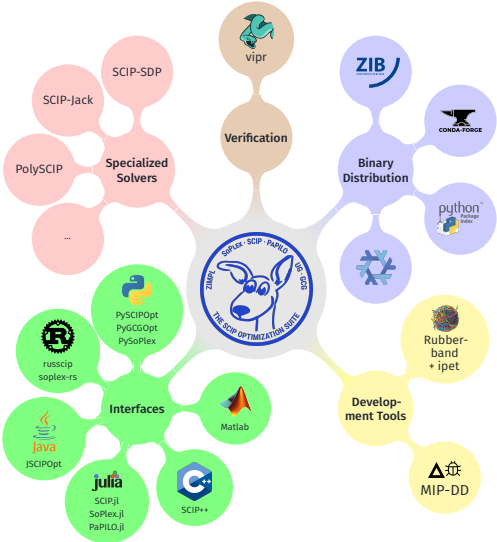
> 240 plugins in the default distribution. SCIP 11 adds a new plugin type: **symmetry handlers**

SCIP is a plugin-based

constraint handlers	34	primal heuristics	67	file readers	28
separators	22	presolvers	19	branching rules	16
expression handlers	13	propagators	12	node selectors	8
event handlers	6	cut selectors	4	relaxators	2

> 240 plugins in the default distribution. SCIP 11 adds a new plugin type: **symmetry handlers**

+ detailed statistics for everything that happens during the search



- ◇ Developed by research groups and industry partners worldwide
- ◇ Annual SCIP Developer Meeting
- ◇ Contributions welcome:
github.com/scipopt



SCIP Developer Meeting 2025, ZIB Berlin

Primal side

- ◇ Feasibility jump heuristic
- ◇ Local search heuristics
- ◇ More LNS

Symmetry

- ◇ New plugin type `symhdlr`: implement your own symmetry handling!

Propagation

- ◇ New propagator for Euclidean distance constraints

Further

- ◇ Improved post-processing of conflict constraints
- ◇ More aggressive restarting, better strong branching allocation

Concurrent solving

- ◇ Solution and variable bounds sharing
- ◇ Focus on primal feasibility

IIS

- ◇ component detection, dynamic batching, new elastic filter
- ◇ $\sim 30\%$ faster

What's new in SCIP 11.0

Primal side

- ◇ Feasibility jump heuristic
- ◇ Local search heuristics
- ◇ More LNS

Symmetry

- ◇ New plugin type `symhdlr`: implement your own symmetry handling!

Propagation

- ◇ New propagator for Euclidean distance constraints

Further

- ◇ Improved post-processing of conflict constraints
- ◇ More aggressive restarting, better strong branching allocation

Concurrent solving

- ◇ Solution and variable bounds sharing
- ◇ Focus on primal feasibility

IIS

- ◇ component detection, dynamic batching, new elastic filter
- ◇ $\sim 30\%$ faster

Already double-digit performance improvement on MIP.

Parallel Racing in SCIP and ReXi

Sequential improvements $\ggg$ parallel tree search

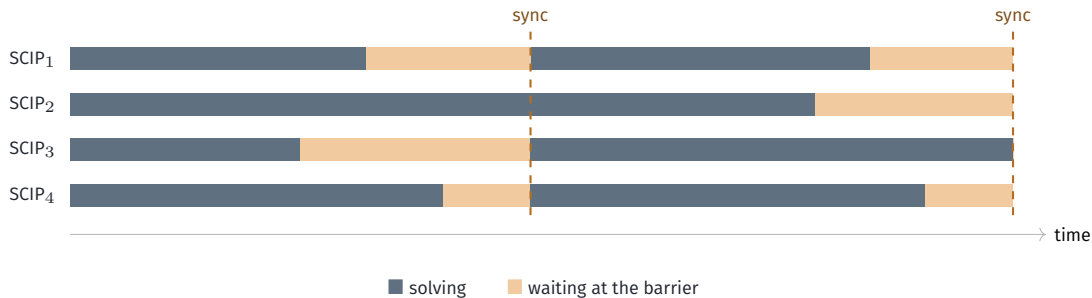
Sequential improvements $\ggg$ parallel tree search

- ◇ **Root bottleneck:** shallow nodes dominate, resources idle until then
- ◇ **Tree shape:** dynamic, unbalanced $\Rightarrow$ load balancing difficult
- ◇ **Ordering matters:** hard to replicate sequential logic without costly control
- ◇ **Data sharing:** cross-tree information sharing is expensive

* See also Ralphs et al. (2018)

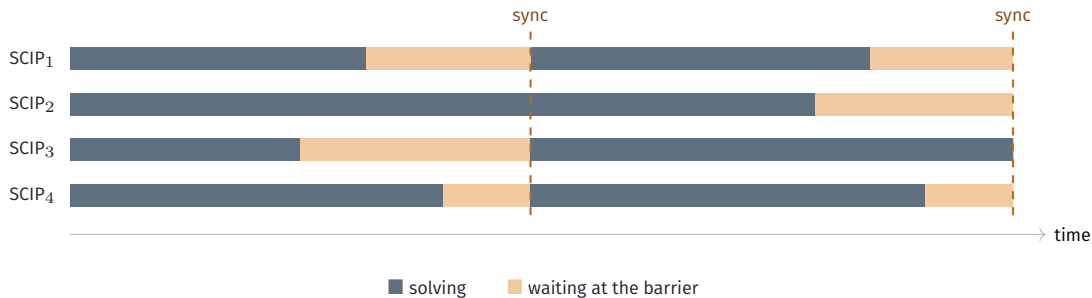
Concurrent SCIP: the classic design

A portfolio of SCIP copies, introduced in SCIP 4.0 (Maher et al. (2017)). Solutions and bounds are exchanged at fixed synchronization points.



Concurrent SCIP: the classic design

A portfolio of SCIP copies, introduced in SCIP 4.0 (Maher et al. (2017)). Solutions and bounds are exchanged at fixed synchronization points.



The exchange is only as fast as the slowest worker.

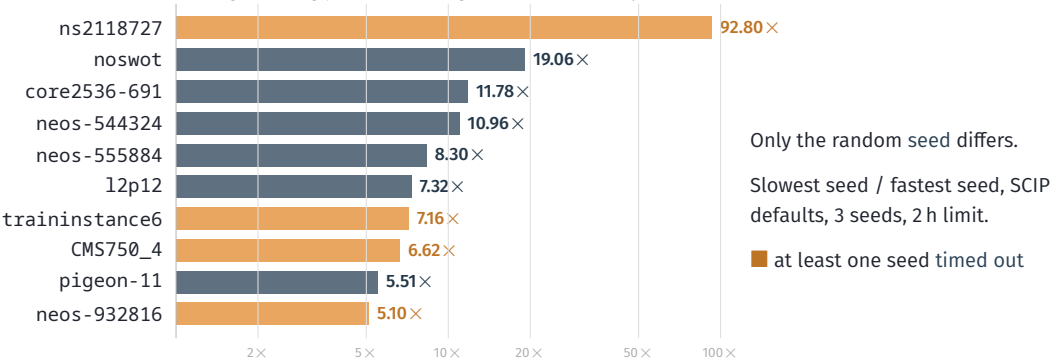
But why would running the *same* problem N times ever help?

Performance Variability (Lodi and Tramontani (2013))

The performance of MIP solvers is subject to unexpected variability, e.g., when changing computing platforms, permuting rows/columns, or adding seemingly neutral changes to the solution process.

Performance Variability (Lodi and Tramontani (2013))

The performance of MIP solvers is subject to unexpected variability, e.g., when changing computing platforms, permuting rows/columns, or adding seemingly neutral changes to the solution process.



Next: A parallel racing framework for SCIP that finds good solutions quickly.

- R** Race. Exploit variability and diversify
- eX** eXchange. Share as much relevant information (solutions, primal bounds, variable bounds) with minimal overhead.
- i** improve. LP-free heuristics, Large Neighborhood Search workers



ReXi Opt

R Race. Exploit variability and diversify

eX eXchange. Share as much relevant information (solutions, primal bounds, variable bounds) with minimal overhead.

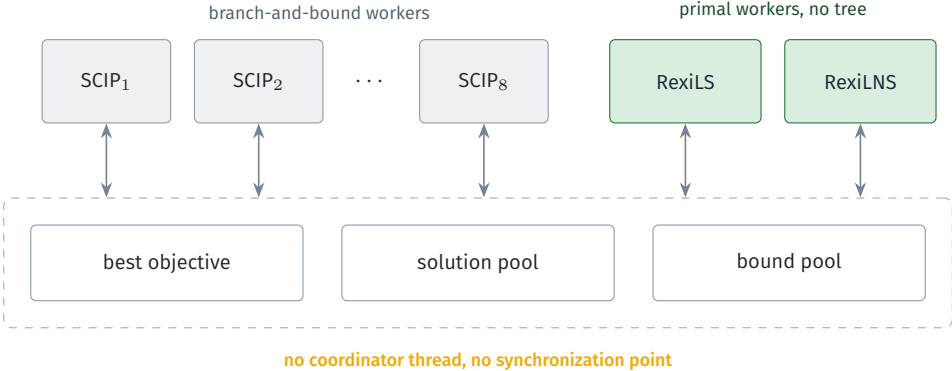
i improve. LP-free heuristics, Large Neighborhood Search workers



Design decisions

- ◇ No coordinator thread, no supervisor, no synchronization points, exchange through shared data structures only
- ◇ Not lock-free, but with minimal contention and very short waiting times

One picture of the whole thing

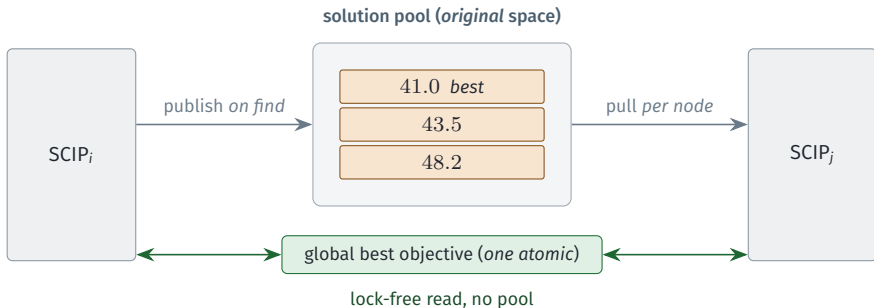


SCIP workers focus on finding good solutions fast and diversifying the search.

- ◇ default SCIP
- ◇ no separation + fast presolve
- ◇ emphasis *feasibility* ($\times 2$, seeds differ)
- ◇ no presolving
- ◇ SAT-like DFS, no LP solves
- ◇ periodic feasibility jump
- ◇ periodic local search
- ◇ local search running perpetually
- ◇ DFS, aggressive restarts + heuristics

eXchange I: solutions and primal bounds

- ◇ Fast lane: the primal bound. One shared atomic, read after *every LP solve* in the cut-and-price loop. Cuts off subtrees and drives reduced-cost fixing.
- ◇ Slow lane: the solutions themselves. Published the moment a worker finds one, pulled only once per node.

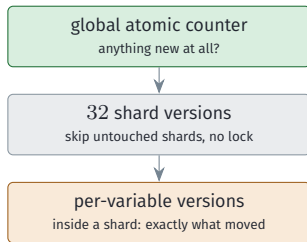


From propagation *and* from presolving.

ReXi's pool

- ◇ ℓ, u , version: three flat arrays, monotone, overwritten in place
- ◇ 32 cache-line-aligned shards, a write locks one

Measured waiting time: $< 1\%$ of runtime.



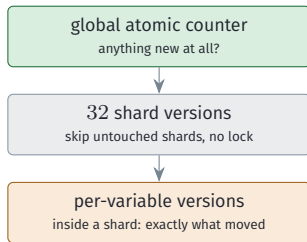
Three-level change detection

From propagation *and* from presolving.

ReXi's pool

- ◇ ℓ, u , version: three flat arrays, monotone, overwritten in place
- ◇ 32 cache-line-aligned shards, a write locks one

Measured waiting time: $< 1\%$ of runtime.



Three-level change detection

The price

Strong dual reductions in one worker only, otherwise the workers jointly cut off every optimum.

RexiLS: LP-free local search (Lin et al. (2025))

- ◇ Owns its thread for the *whole* solve window
- ◇ Restarts from the pool incumbent when it stalls
- ◇ Matrix mirrored once into CSC + CSR: a column scan is two contiguous reads, not three pointer chases

RexiLNS: neighborhoods built from the race

- ◇ Mutation: fix each integer to its rounded reference value, with probability α
- ◇ Crossover: the agreement set of two solutions from *different* workers
- ◇ Pool still empty? Take the local search's least-violating point

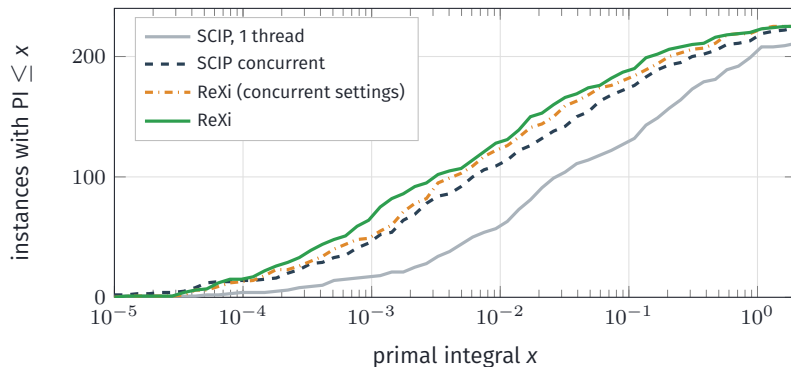
Related: evolutionary polishing in CPLEX (Rothberg (2007)), dedicated LNS workers in CP-SAT (Perron and Didier (2023)).

- ◇ 233 instances: MIPLIB 2017 benchmark (Gleixner et al. (2021)) without the known-infeasible ones
- ◇ 600 s time limit, 96 GB memory, AMD EPYC 9B45, nodes used exclusively
- ◇ 10 threads, one instance per node
- ◇ Time and primal integral (PI) as shifted geometric means, shifts 1 s and 0.001

	threads	feas.	opt. found	opt. proven	time [s]	PI
SCIP	1	211	119	95	201.2	0.0568
SCIP concurrent	10	224	154	119	149.6	0.0178
ReXi (concurrent settings)	10	225	158	123	142.8	0.0134
ReXi	10	225	160	124	135.3	0.0107

0.0568 $\rightarrow$ 0.0107: more than $5\times$ on the primal integral.

Is it just a few instances?



For a threshold x , how many of the 233 instances reach a primal integral of at most x . Up and to the left is better.

Which exchange actually pays?

Switch off one mechanism at a time, ten-thread portfolio otherwise unchanged.

	feas.	opt. found	opt. proven	time [s]	PI
ReXi	225	160	124	135.3	0.0107
no solution exchange	224	150	115	156.0	0.0146
no bound exchange	224	157	124	137.6	0.0118

Where we stand

Against the build submitted on Michael Bussieck's
benchmark:  G A M S
plato.asu.edu/ftp/mipfeas.html

	opt. found	opt. proven	time [s]	PI
ReXi	160	124	135.3	0.0107
submission ^a	148	116	151.7	0.0164

RexiLS, bound sharing and RexiLNS:

PI -35% , $+12$ optima found, $+8$ proven, -11% time.

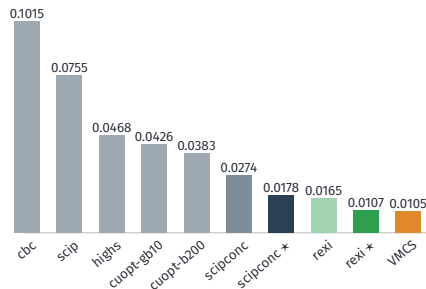
* VMCS commercial solvers at defaults

^arerun independently



ReXi Opt

Geom. Mean (shift = 0.001)



- ◇ Race, eXchange, improve pays off. Large improvements on primal feasibility
- ◇ Open: More threads, more specialized workers
- ◇ Open: GPU-accelerated workers

Thank you!

mexi@zib.de



For the latest updates:



SCIP on LinkedIn



<https://www.scipopt.org/blog/>

- Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(2):443–490, 2021.
- Peng Lin, Shaowei Cai, Mengchuan Zou, and Jinkun Lin. Local-mip: Efficient local search for mixed integer programming. *Artificial Intelligence*, 348:104405, 2025. ISSN 0004-3702. doi: 10.1016/j.artint.2025.104405.
- Andrea Lodi and Andrea Tramontani. Performance Variability in Mixed-Integer Programming. In Huseyin Topaloglu, J. Cole Smith, and Harvey J. Greenberg, editors, *Theory Driven by Influential Applications*, pages 1–12. INFORMS, September 2013. ISBN 978-0-9843378-4-2. doi: 10.1287/educ.2013.0112.

Stephen J. Maher, Tobias Fischer, Tristan Gally, Gerald Gamrath, Ambros Gleixner, Robert Lion Gottwald, Gregor Hendel, Thorsten Koch, Marco Lübbecke, Matthias Miltenberger, Benjamin Müller, Marc Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Dieter Weninger, Jonas T. Witt, and Jakob Witzig. The SCIP optimization suite 4.0. Technical Report 17-12, ZIB, Takustr. 7, 14195 Berlin, 2017.

Laurent Perron and Frédéric Didier. Cp-sat, 2023. URL https://developers.google.com/optimization/cp/cp_solver/.

Ted Ralphs, Yuji Shinano, Timo Berthold, and Thorsten Koch. Parallel solvers for mixed integer linear optimization. In *Handbook of parallel constraint reasoning*, pages 283–336. Springer, 2018.

Edward Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.