

CalSim3 Solver Benchmarking

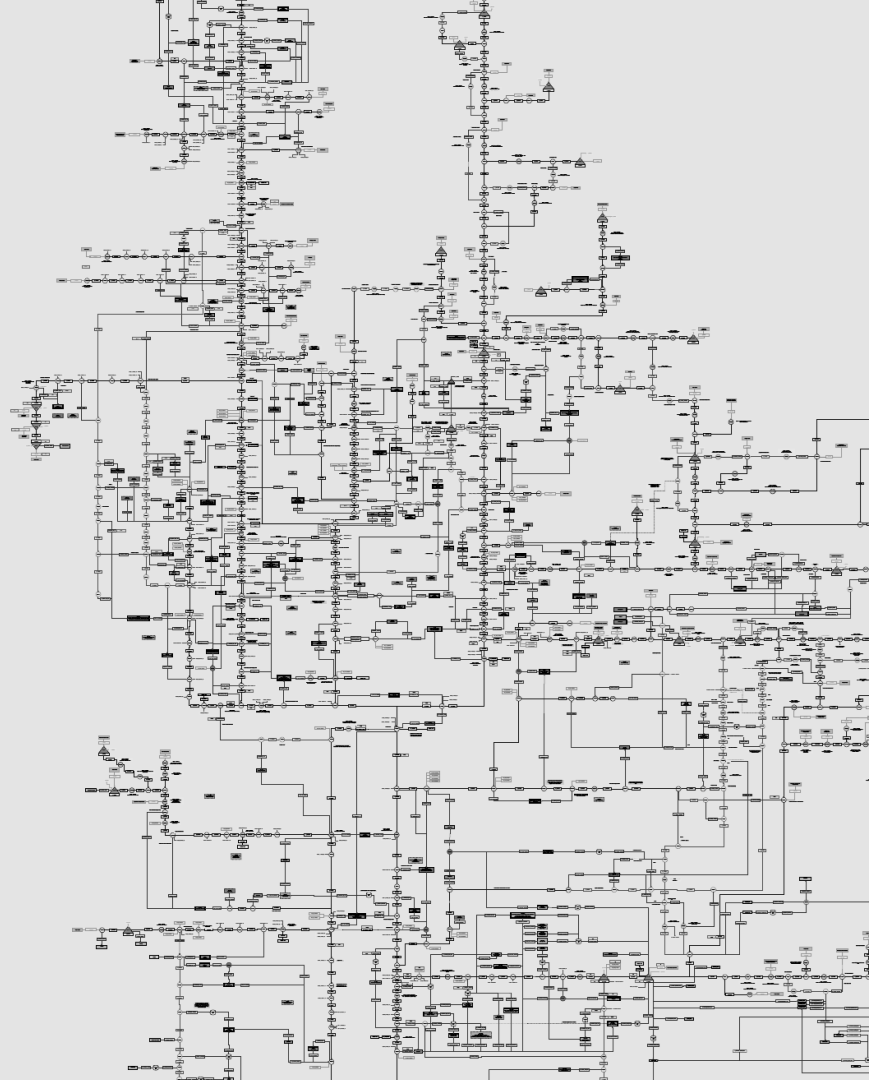
CWEMF 2026 – Session 7. Updates in CalSim Development



CALIFORNIA DEPARTMENT OF
WATER RESOURCES

Jon Herman
Samarth Singh

Yiwei Cheng
Zach Roy
Nicole Osorio
Hamed Zamanisabzi
Jamel Lehyan
Bing Li



CalSim3 Model

- Statewide long-term planning model jointly developed by DWR/USBR
- Mixed-integer linear programming (MILP), monthly timestep
- 38 optimization cycles up to ~10,000 decision variables and constraints

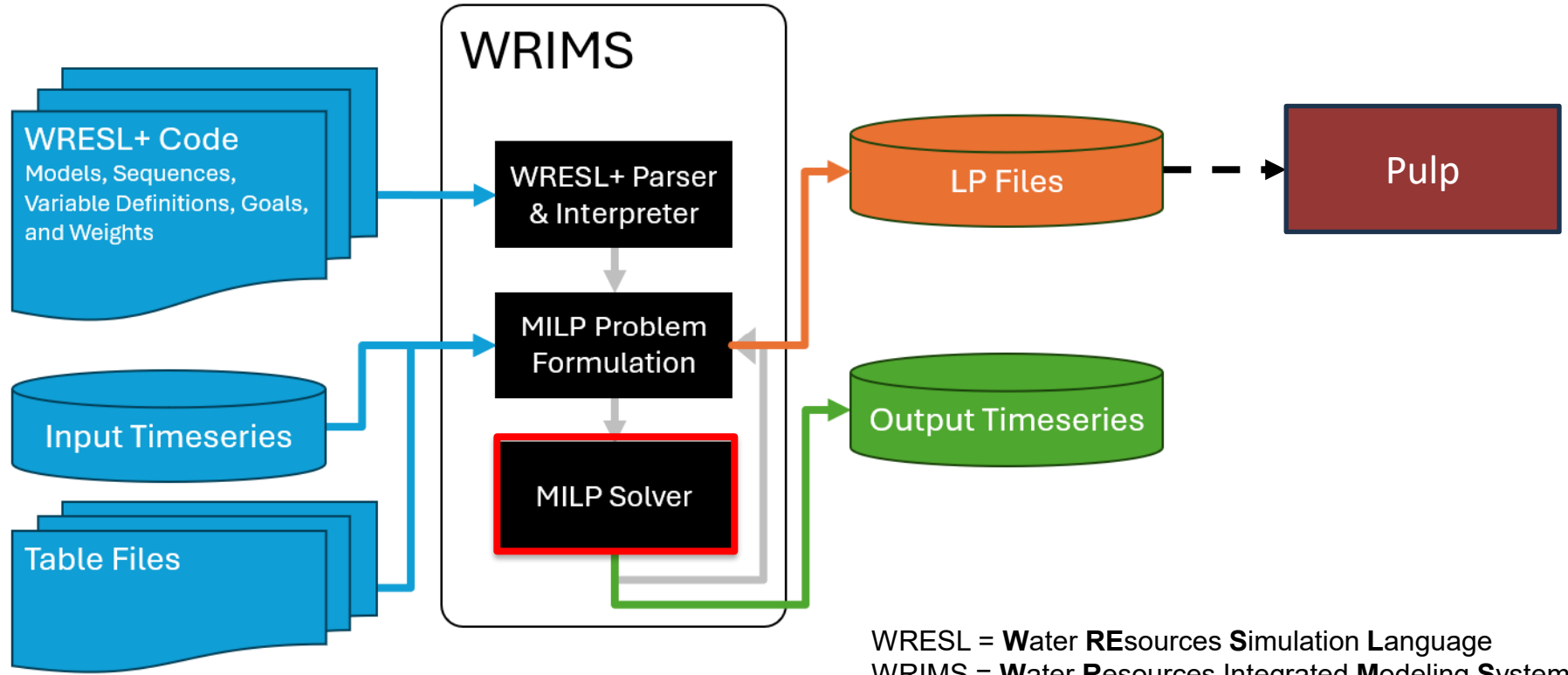
Typical model run (1922-2021) takes
~4 hours on a standard PC, with about
half of that time in the solver.

Can this be faster?



[Figure credit: CS3 Schematic / DWR]

The solver is only one part of the WRIMS engine



WRESL = **W**ater **R**esources **S**imulation **L**anguage
WRIMS = **W**ater **R**esources **I**ntegrated **M**odeling **S**ystem
MILP = **M**ixed **I**nteger **L**inear **P**rogramming

Solvers available in WRIMS

Solver	License	Performance	Community Support	Typical Use Case in CalSim
CBC	Free	Moderate	Active - Open Source (COIN-OR)	Default (DWR/USBR)
Gurobi	Commercial /Free Academic	Very Fast	Commercial Support Online Forums	In Testing Phase
XA	Commercial	Good	Commercial Support	In Use (DWR/USBR)

Research Questions

- How do solver runtimes compare in (a) pure solver time and (b) total WRIMS runtime?
- Are Gurobi/XA consistent with the default CBC solutions? (decision variables and objectives)
- Are the findings robust to solver parameters: feasibility tolerance and number of threads?

Experiments: DCR23 (1922-2021)



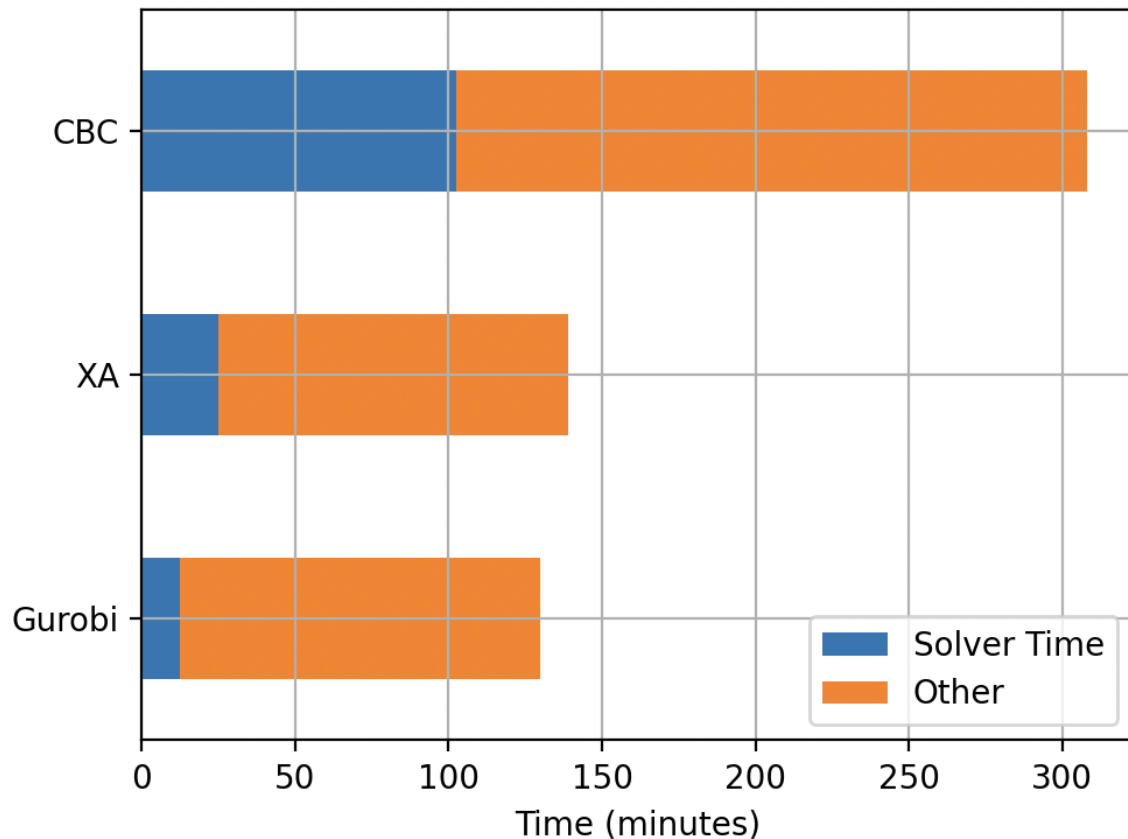
Experiment	Historical (Adj)	CC50	CC75	CC95
WRIMS 3 (Windows PC)	Gurobi, CBC, XA	Gurobi, CBC	Gurobi, CBC	Gurobi, CBC
PuLP (Linux cluster using exported LP files)	Gurobi, CBC	--	--	--

- Compare decision variables, objectives, and runtimes across all runs
- Parameter settings test: feasibility tolerance and number of threads

Windows PC: Intel i7-7700 CPU @ 3.60 GHz, serial

UC Davis HPC2 cluster: AMD EPYC 7532 @ 2.4 GHz, 1-8 threads

Results: Total WRIMS runtime (DCR23 1922-2021)



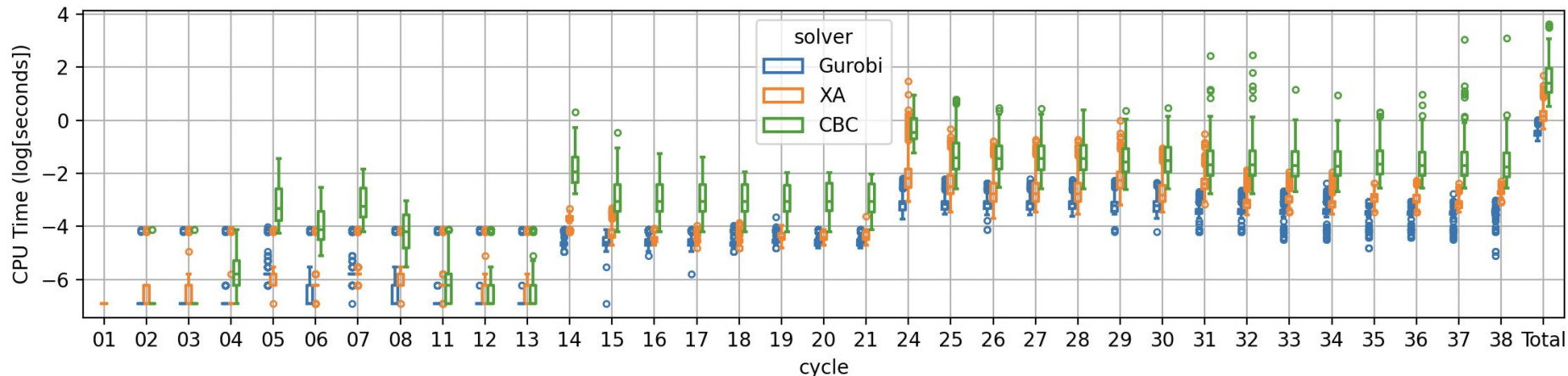
- CBC is much slower, and the Java implementation adds more time to WRIMS
- Gurobi/XA total time is similar – the difference is likely in the problem construction
- Gurobi spends only 12 out of 130 minutes in the solver

WRIMS timing comparison (scaled, Gurobi = 1.0)

	Gurobi	CBC	XA
Python (solver only)	1.0	3.9	n/a
WRIMS (solver only)	1.0	8.2	1.9
WRIMS (total)	1.0	2.4	1.07

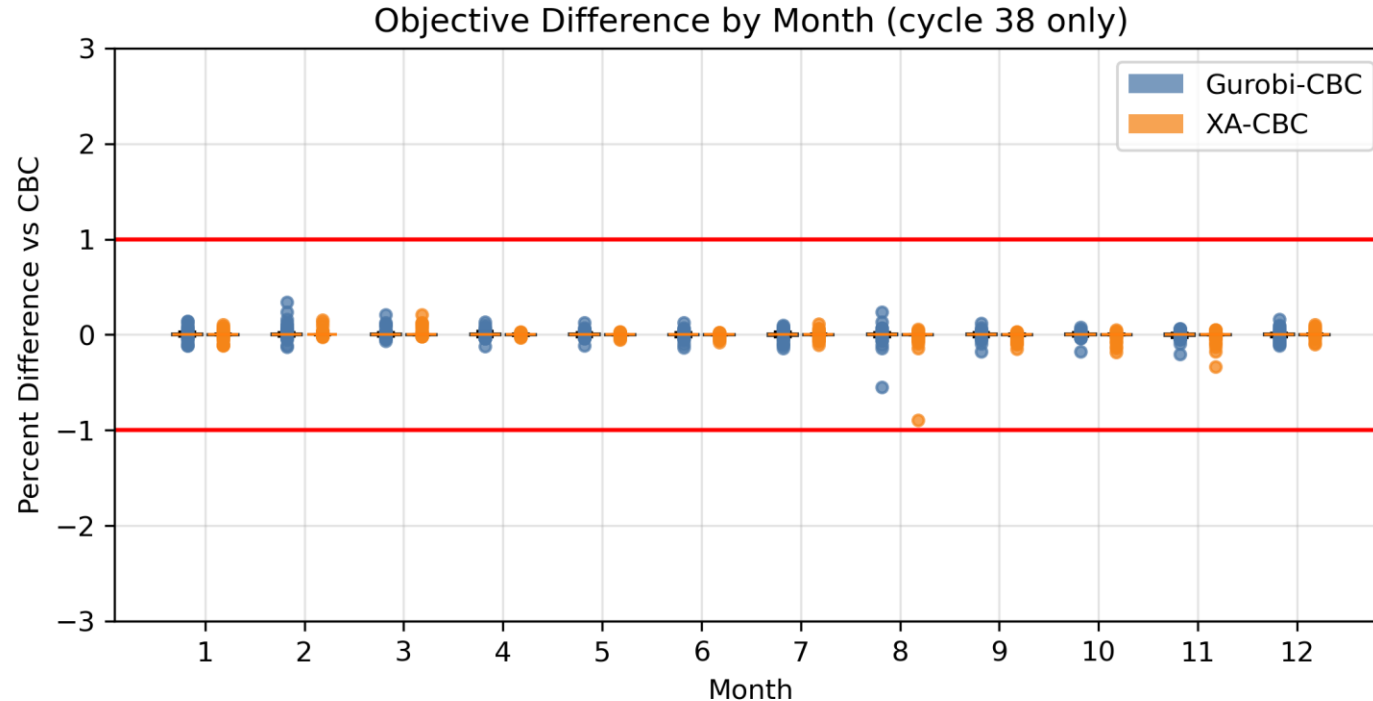
- Python timing (exported LP files) is the best measure of the solver alone
- WRIMS: different solver classes and versions
- WRIMS does a lot of other work outside of the solver

Timing comparison by cycle



- Gurobi is faster and more reliable across all cycles
- Most runtime comes from cycles 24-38 (larger # decision variables)
- Runtime ratios $\sim 2\times$ XA, $\sim 6-8\times$ CBC

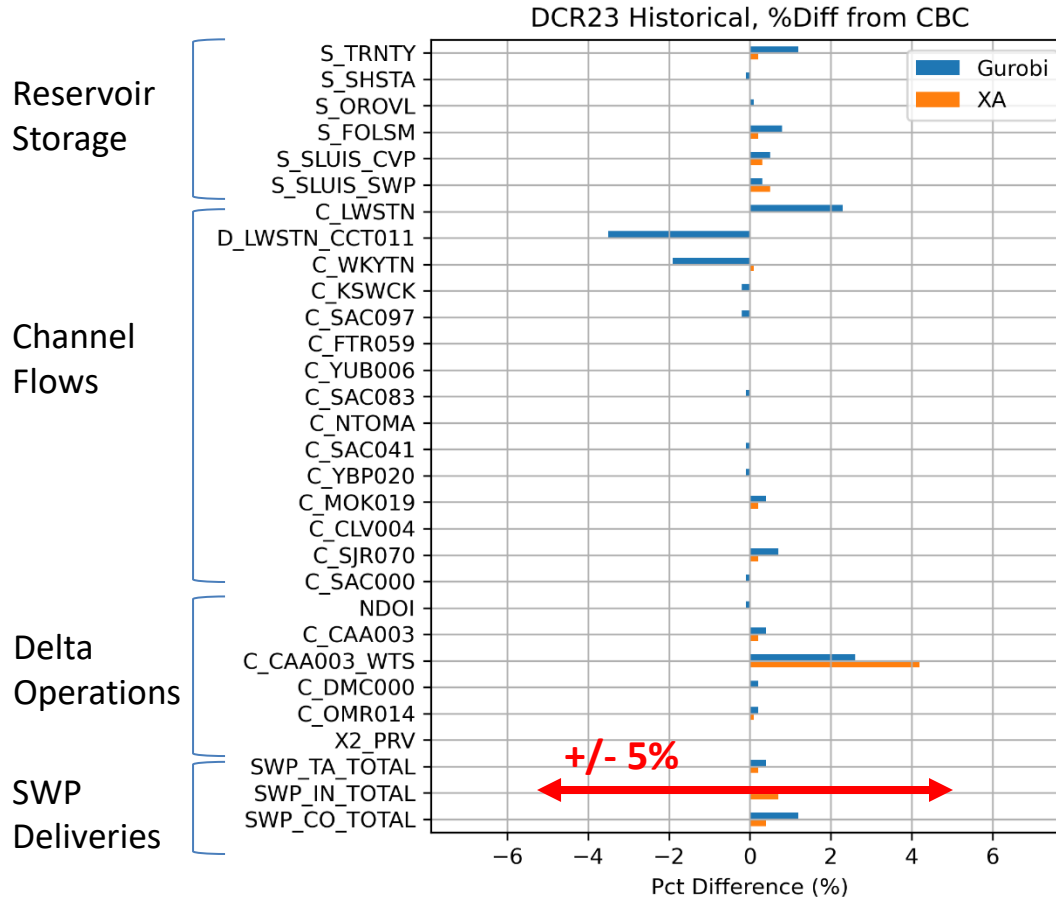
WRIMS objective comparison (1922-2021)



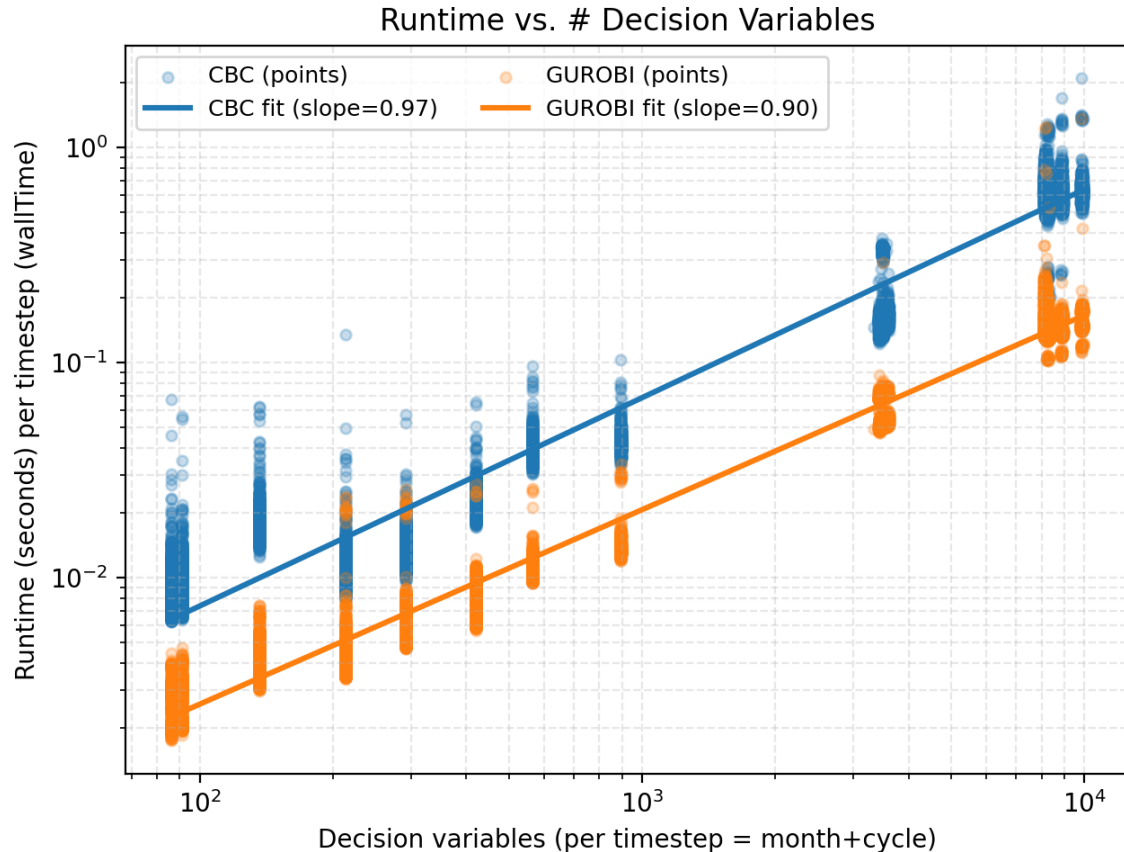
- 100 points per month. All objective values within $\pm 1\%$

Decision variables

Good match with CBC in historical and climate change scenarios

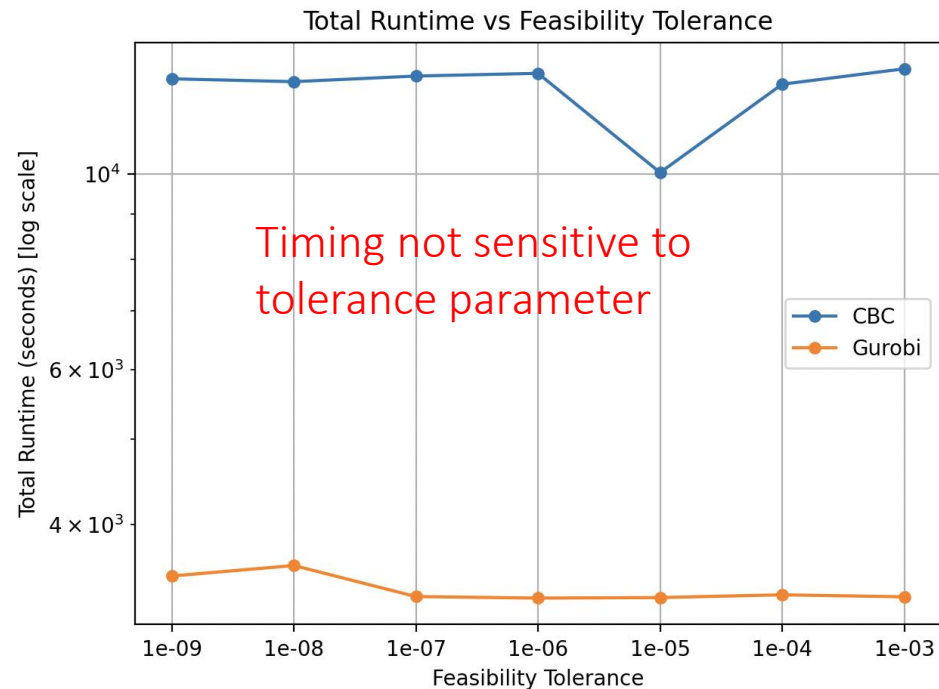
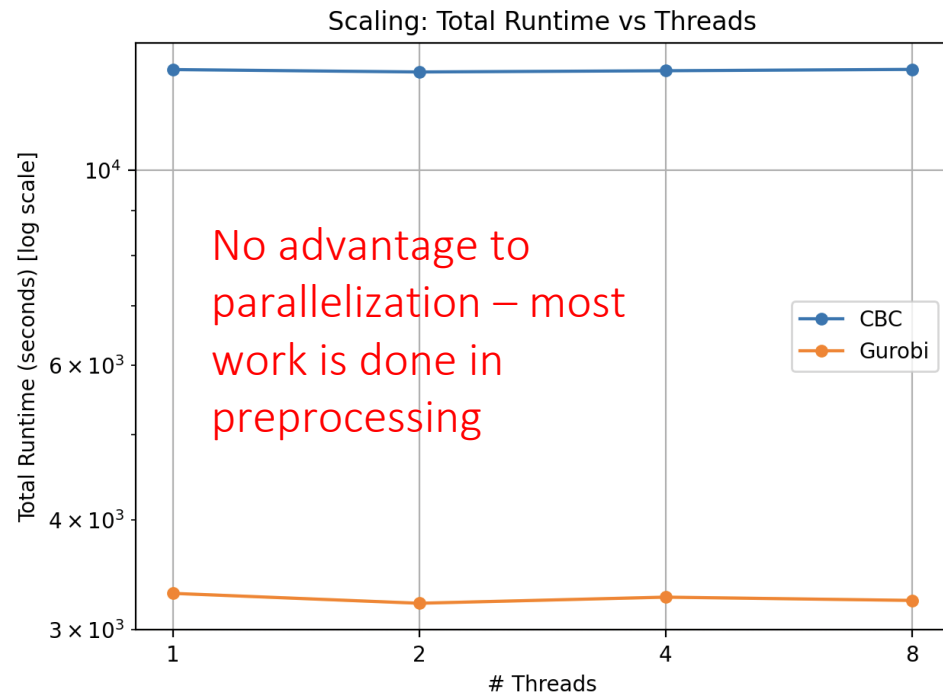


Scaling with problem size (Python)



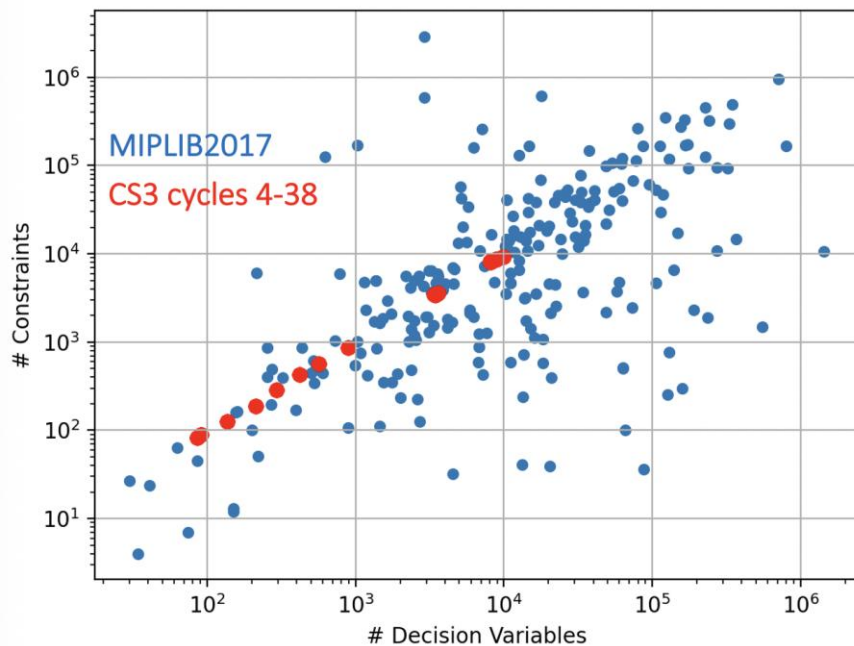
- Each cycle contains 1200 points (months)
- Gurobi speedup of 3-4x across cycles
- Best-fit slope shows slightly better scaling (0.90 vs. 0.97)

Parameter sensitivity: # threads and tolerance (Python)



Comparison to MIPLIB Benchmarks (2021)

- 240 mixed-integer benchmark problems from operations research



- Gurobi solves the most # problems (201/240) and is $\sim 8x$ faster than CBC (geometric mean across problems)

1 thr	CBC	GLPK	LP_SOL	MATLAB	SCIP	Gurobi	COPT
unscal	2107	5044	5335	3301	1100	245	1029
scaled	8.59	20.5	21.7	13.5	4.48	1	4.19
solved	89	23	20	63	125	201	132

Summary: CalSim3 Solver Benchmarking

- Gurobi is $\sim 4\times$ faster than CBC for CalSim3 problems
- The WRIMS implementation is $\sim 8\times$ faster. The solver is only one component of the total runtime.
- DVs and objectives are a good match with CBC for the historical and climate change scenarios
- Timing is not sensitive to feasibility tolerance or multi-threading

Opportunities for future development

- Time efficiency outside the solver
 - Open-source community effort
- Abstract solver implementation
- How to leverage modern solver capability?
 - Timestep, cycle structure, nonlinear constraints
 - Automatic feasibility relaxation