



Recent Advances In The New Mosek 8

16-November-2016

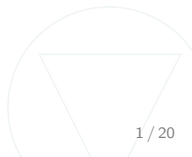
Erling D. Andersen

www.mosek.com





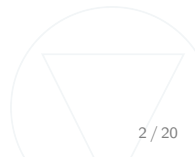
- Presolve
 - Reduce memory requirements in the eliminator.
 - Conic presolve.
- Conic optimizer
 - Improve numerical stability particularly for SDOs.
 - Added a (auto) dualizer for conic quadratic problems.
- Convex quadratic optimizer
 - Use the conic optimizer internally.
- Mixed-integer optimizer.
 - Improved performance.
- Fusion.
 - Added a C++ version.
 - A lot of polishing and bug fixing.
- Optimization server.
- Conda availability.





- More aggressive problem scaling.
- Reworked formulas.
- Improved stopping criterion.
- Improved linear algebra.
- Automatic and transparent dualizer (not available for SDOs yet).
- Better ill posed problem handling e.g.

$$\begin{array}{ll} \text{minimize} & x_3 \\ \text{subject to} & x_1 = 0, \\ & 2x_1x_2 \geq x_3^2. \end{array}$$





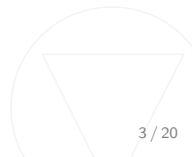
Convex quadratic optimization model

$$\begin{aligned} & \text{minimize} && \frac{1}{2}x^T Q_0^T x + c^T x \\ & \text{subject to} && \frac{1}{2}x^T Q_i^T x + a_i x \leq b_i, \quad i = 1 \dots, m. \quad (\text{QO}) \end{aligned}$$

where:

- Symmetry: $Q_i = Q_i^T, \quad i = 1, \dots, m.$
- Convexity: $Q_i \succeq 0.$

Hence, Q_i should be **positive semidefinite**.

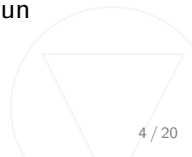




MOSEK v8 convert (QO) to a (CQO) internally:

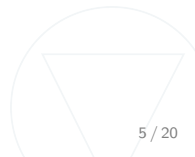
$$\begin{aligned} \text{minimize} \quad & t_0 + c^T x \\ \text{subject to} \quad & t_i + a_i : x = b_i, \quad i = 1, \dots, m, \quad (\text{CQO}) \\ & L_i^T x - f_i = 0, \\ & z_i = 1, \\ & 2z_i t_i \geq \|f_i\|^2. \end{aligned}$$

- Conversion to conic form is transparent.
- Leads to faster and more robust solution.
- Makes solution of convex quadratic problem run-to-run deterministic when multiple threads is employed.





- Improved presolve.
- Presolve repeated after cutting plane generation if the optimizer deems it worthwhile.
- Improved heuristics.
- New cutting planes: Knapsack cover cuts.





Example:

$$\begin{array}{ll} \text{maximize} & \mu^T x \\ \text{subject to} & e^T x = w + e^T x^0, \\ & [\gamma; G^T x] \in \mathcal{K}_q^{n+1}, \\ & x \geq 0. \end{array}$$



```
def BasicMarkowitz(n,mu,GT,x0,w,gamma):
    with Model("Basic Markowitz") as M:

        # Redirect log output from the solver to stdout for debugging.
        # if uncommented.
        M.setLogHandler(sys.stdout)

        # Defines the variables (holdings). Shortselling is not allowed.
        x = M.variable("x", n, Domain.greaterThan(0.0))

        # Maximize expected return
        M.objective('obj', ObjectiveSense.Maximize, Expr.dot(mu,x))

        # The amount invested must be identical to initial wealth
        M.constraint('budget', Expr.sum(x), Domain.equalsTo(w+sum(x0)))

        # Imposes a bound on the risk
        M.constraint('risk', Expr.vstack( gamma,Expr.mul(GT,x)),
                    Domain.inQCone())

    M.solve()
```

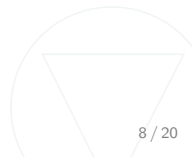



General:

- Bug fixes.
- Polishing.
- Improved expression generators and performance.
- Support for C++ on Linux and MAC OS.
- Windows C++ support is waiting for a bug fix in MSC.

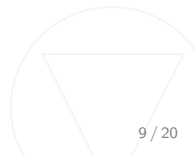
Python:

- Support for Python 3.4+ and Pypy/2.7.
- Use of Numpy arrays are default.



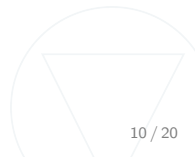


- Solve problems remotely with MOSEK using the optimization server.
- Submit problems, get solution through the web interface.





- Install the latest MOSEK using Anaconda
 - `conda install -c mosek mosek=8.0.43`
- Support for Anaconda Python 2.7, 3.4, and 3.5





- Hardware: Intel based server.
- Software: Linux. MOSEK v7.1.0.58 and v8.0.0.42.
- Threads: 8 threads is used in test to simulate a typical user environment.
- All timing results t are in wall clock seconds.
- A small problem: Fastest optimizer has $t \leq 6$.
- A medium problem: Fastest optimizer has $t \leq 60$.
- Test problems: Public (e.g `cblib.zib.de`) and customer supplied.

For each problem we compute

$$r = \frac{t_8 + 0.01}{t_7 + 0.01}$$

where t_8 and t_7 is taken to solve a problem for version 8 and 7 respectively. The geometric average of the r s is shown.



	small		medium		large		fails'	
	7	8	7	8	7	8	7	8
Num.	887	887	127	127	30	30	37	18
Firsts	579	535	20	107	2	28		
Total time	1985.914	794.085	3369.599	1678.912	22255.519	10794.724		
G. avg. r		0.88		0.50		0.59		

- Version 8 has fewer failures.
- Version 8 has more firsts.
- Version 8 is at least 40% faster on average for medium to large problems.

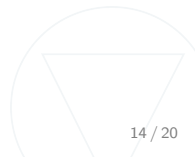


	small		medium		large		fails'	
	7	8	7	8	7	8	7	8
Num.	111	111	55	55	22	22	62	8
Firsts	35	94	0	55	0	22		
Total time	228.043	124.442	2484.902	1621.460	7077.588	4111.834		
G. avg. r		0.73		0.65		0.59		

- Version 8 is more stable and has the most firsts.
- Version 8 is at least 40% faster on average for medium to large problems.

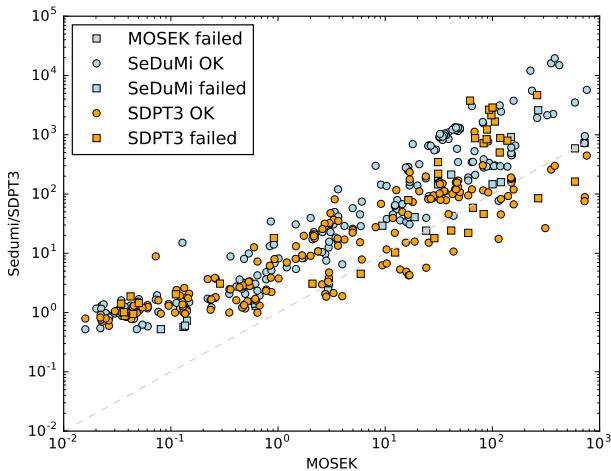


- We use a subset of Mittelmann's benchmark-set + customer provided problems.
- MOSEK on 4 threads, MATLAB up to 20 threads.
- Problems are categorized as **failed** if
 - MOSEK returns unknown solution status.
 - SeDuMi returns `info.numerr==2`.
 - SDPT3 returns `info.termcode` $\notin [0, 1, 2]$ and `norm(info.dimacs, Inf) > 1e-5`.



Semidefinite problems

Comparison with SeDuMi and SDPT3



Solution time for MOSEK vs SeDuMi/SDPT3 on 234 problems.



	small			medium		
	MOSEK	SeDuMi	SDPT3	MOSEK	SeDuMi	SDPT3
Num.	127	127	127	63	63	63
Firsts	116	1	10	47	0	16
Total time	218.2	1299.5	843.6	2396.0	32709.8	5679.5

	large			fails		
	MOSEK	SeDuMi	SDPT3	MOSEK	SeDuMi	SDPT3
Num.	44	44	44	6	27	47
Firsts	31	0	13			
Total time	9083.8	119818.1	35268.2			

- MOSEK is fastest on average with fewest failures.
- SeDuMi is almost always slower than MOSEK.



	small		medium		large		fails'	
	7	8	7	8	7	8	7	8
Num.	466	466	19	19	8	8	28	34
Firsts	336	260	3	16	2	6		
Total time	44514.992	197.182	25958.112	415.943	5447.486	3167.290		
G. avg. r		0.97		0.27		0.64		

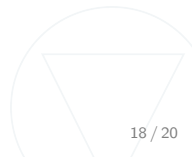
- Version 8 has a few more failures and slightly slower on average for small problems.
- Version 8 is faster for medium to large problems.



- Timeout set to 3600 seconds.
- Optimality gap set to 0.0.

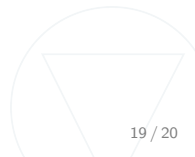
Version	7	8
Num.	181	181
Firsts	30	58
Solved	123	128
Total time	46546	26162
G. avg. r		0.65

- Version 8 is significantly faster.
- Solves a few more problems to optimality.





- MOSEK version 8 is a significant improvement over version 7.
- Particularly the robustness of semidefinite optimizer has been improved dramatically.
- On average a 20% to 40% reduction in the solution time can be expected for quadratic and conic problems





With MOSEK v8 SDO starts getting practible!

