



Recent advances in MOSEK

AAU, 22 of August 2016

e.d.andersen@mosek.com

www.mosek.com





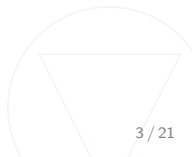
- ① An introduction to MOSEK.
- ② Improvements in (the latest and greatest) version 8.
- ③ Some benchmark results.
- ④ Thoughts about embedded usage of MOSEK.



- MOSEK develops and sells optimization software.
- Versions: 1 (1999), 7 (2014), 8 (2016).
- Linear and convex quadratic (+ mixed-integer).
- Conic quadratic optimization (+ mixed-integer).
- Semidefinite optimization.
- Convex(functional) optimization.
- Optimizer API: C, JAVA, .NET, Python.
- Fusion API: C++, JAVA, MATLAB, .NET, Python.
- AMPL, GAMS, Julia, MATLAB, R interfaces and more.
- Free for academic use. See <http://www.mosek.com>.

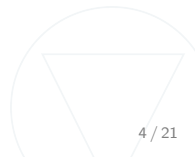


- Presolve
 - Reduce memory requirements in the eliminator.
 - Conic presolve.
- Conic optimizer
 - Improve numerical stability particularly SDOs.
 - Added a (auto) dualizer for conic quadratic problems.
 - Improved sequential and parallel performance.
- Convex quadratic optimizer.
 - Use the conic optimizer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





- More aggressive problem scaling.
- Reworked formulas.
- Improved stopping criterion.

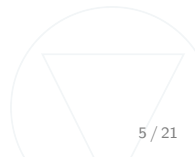




The problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \\ & x \in K. \end{array}$$

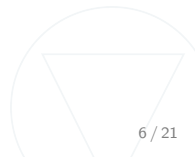
where K is a Cartesian product of symmetric cones.





$$\begin{aligned}Ax - b\tau &= 0, \\A^T y + s - c\tau &= 0, \\-c^T x + b^T y - \kappa &= 0, \\x, s &\in K, \\ \tau, \kappa &\geq 0.\end{aligned}$$

which is solved using an interior-point method inside **MOSEK** .





If

$$\begin{aligned} \left\| A \frac{x^k}{\tau^k} - b \right\|_{\infty} &\leq \epsilon_p (1 + \|b\|_{\infty}), \\ \left\| A^T \frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_{\infty} &\leq \epsilon_d (1 + \|c\|_{\infty}), \text{ and} \\ \min \left(\frac{(x^k)^T s^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) &\leq \epsilon_g \max \left(1, \frac{\min(|c^T x^k|, |b^T y^k|)}{\tau^k} \right), \end{aligned}$$

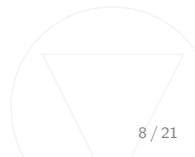
then $\frac{(x^k, y^k, s^k)}{\tau^k}$ is declared a primal-dual optimal solution.



If

$$\begin{aligned} \left\| A^T y^k + s^k \right\|_{\infty} &\leq \epsilon_d \|y\|_{\infty}, \\ \epsilon_i b^T y^k &> \|b\|_{\infty} \left\| A^T y^k + s^k \right\|_{\infty}, \end{aligned}$$

then the problem is declared primal infeasible and (y^k, s^k) is a certificate.

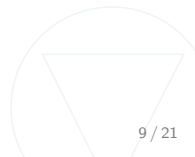




If

$$\begin{aligned} \|Ax^k\|_\infty &\leq \epsilon_p \|x\|_\infty, \\ -\epsilon_i c^T x^k &> \|c\|_\infty \|Ax^k\|_\infty, \end{aligned}$$

then the problem is declared dual infeasible and x^k is a certificate.





If

$$\begin{aligned} \max \left(|c^T x^k|, \|Ax^k\|_\infty \right) &\leq \epsilon_p \max(1, \|x\|_\infty), \\ \max \left(|b^T y^k|, \|A^T y^k + s^k\|_\infty \right) &\leq \epsilon_d \max(1, \|y\|_\infty), \end{aligned}$$

then the problem is primal illposed if

$$\epsilon_i \|y^k\|_\infty \max(1, \|b\|_\infty) > \max \left(|b^T y^k|, \|A^T y^k + s^k\|_\infty \right)$$

and dual illposed if

$$\epsilon_i \|x^k\|_\infty \max(1, \|c\|_\infty) > \max \left(|c^T x^k|, \|Ax^k\|_\infty \right).$$

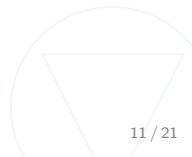


Challenges:

- A computer has many cores.
- Parallelization using native threads is cumbersome and error prone.
- Employ a parallelization framework e.g. Cilk or OpenMP.

Other issues:

- Exploit caches.
- Do not overload the memory bus.
- Threading overhead (not fine grained).
- Increasing return to scale on BLAS calls.





- Extension to C and C++.
- Has a runtime environment that execute tasks in parallel on a number of workers.
- Handles the load balancing.
- Allows nested/recursive parallelism e.g.
 - Parallel dense matrix mul. within parallelized sparse Cholesky.
 - Parallel IPM within B&B.
- Is run to run deterministic.
 - Care must be taken in floating point computations.
- Supported by the Intel C, Gcc, Clang.



The dense level 3 BLAS syr_k operation does

$$C = AA^T.$$

Parallelized version using Cilk:

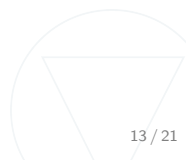
If C is small

$$C = AA^T$$

else

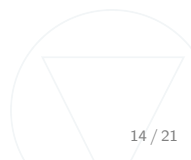
```
cilk_spawn  C21 = A2:A1T  gemm
cilk_spawn  C11 = A1:A1T  syrk
cilk_spawn  C22 = A2:A2T  syrk
cilk_sync
```

Usage of recursion is allowed!





- Cilk is easy to learn i.e. 3 new keywords.
- Nested/recursive parallelism is allowed.
- Useful for both sparse and dense matrix computations.
- Efficient parallelization is nevertheless hard.



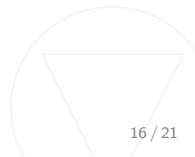


- Linear algebra parallelized using Cilk.
- Rewritten Cholesky using Intel MKL BLAS/LAPACK library.
- Rewritten dense column handling and Schur computations.
- Tuned graph partitioning based ordering.





- Fact: Worthwhile to input the dual as the primal occasionally.
- Or use the new dualizer for conic quadratic problems.
- Dualizes before presolve.
- Is transparent.
- Has a heuristic for when to dualize: The hard part.





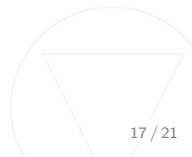
The quadratic optimization model

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

Assumptions:

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

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



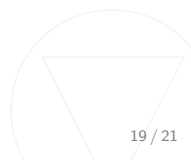


MOSEK v8 convert (QO) to a (CQO) internally:

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



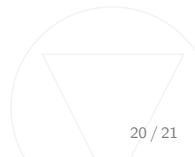
- Conversion is transparent.
- Both problems solves in the same worst case complexity using an interior-point method.
- No bad duality states is introduced in the conic reformulation ART [1].
- Formulation is bigger.
- But may be sparse (think dense low rank Q s).
- **MOSEK** v8 makes the conversion automatically.
- Not fool proof in the semidefinite case. (Similar to linear dependency check.)
- Recommendation: The user should do it.





Environment:

- Hardware: Intel based server.
- Software: Linux. MOSEK v7.1.0.51 and v8.0.0.24(beta).
- Threads: 4.
- Small: Fastest optimizer $t \leq 6$.
- Medium: Fastest optimizer $t \leq 60$.





	small		medium		large		fails'	
	7	8	7	8	7	8	7	8
Num.	953	953	56	56	17	17	37	15
Firsts	558	740	16	40	2	15		
Total time	2654.850	869.345	2213.579	1248.068	12719.809	5857.705		
G. avg. r		0.65		0.73		0.54		

- Version 8 has fewer failures.
- Version 8 has more firsts.
- Version 8 is at least 20% faster on average.



	small		medium		large		fails'	
	7	8	7	8	7	8	7	8
Num.	110	110	52	52	22	22	63	5
Firsts	38	100	4	48	5	17		
Total time	184.841	109.944	1722.792	1352.167	5182.408	3576.445		
G. avg. r		0.69		0.76		0.72		

- Version 8 is much more stable.
- Version 8 has more firsts.
- Version 8 is at least 20% faster on average.



	small		medium		large		fails'	
	7	8	7	8	7	8	7	8
Num.	443	443	11	11	4	4	21	36
Firsts	324	268	2	10	0	4		
Total time	418.817	143.300	21248.208	223.253	1996.698	1479.297		
G. avg. r		0.97		0.21		0.68		

- Version 8 has **more** failures on these problems.
- Version 8 has more firsts for large problems.
- Version 8 is on average faster.



Embedded requests/usage of MOSEK:

- Robots.
- Cars.
- Control a cooling system in containers. (MIP)

Requirements to build and run:

- C compiler + Python 2.7.
- Dynamic memory allocation.
- Double precision math (+ good BLAS library).

Answer:

- Stripped down low overhead interface.
- No presolve, no dualization etc.

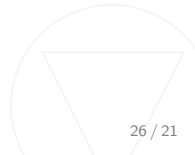


Evaluation:

- Is there a sufficiently big market for an embedded MOSEK?



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

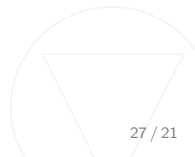




E. D. Andersen, C. Roos, and T. Terlaky.

Notes on duality in second order and p-order cone optimization.

Optimization, 51(4):627–643, 2002.





Thank you!

e.d.andersen@mosek.com

www.mosek.com

