Mathematical Programming manuscript No.
(will be inserted by the editor)

Michal Kočvara · Michael Stingl

Erratum: On the solution of large-scale SDP problems by the modified barrier method using iterative solvers

Received: date / Revised version: date

In the article [1] published in *Mathematical Programming* in 2007 we described a method for the solution of large scale linear semidefinite programming problems. The algorithm was based on a generalization of the augmented Lagrangian algorithm. The new element described in the article was the use of preconditioned conjugate gradient method for the solution of linear systems appearing in the Newton method, used to solve unconstrained optimization problems in the major iterations of the algorithm. We provided complexity analysis of the algorithm and performed a series of numerical tests. The tests showed that the new code was particularly successful for problems with a large number of variables and a medium size of the matrix constraints.

Since then, several articles and preprints have been published, describing similar algorithms and comparing their computational efficiency with, among others, the tests published in [1]; see, e.g., [2–4]. These new algorithms and codes have shown superior behavior, in particular, for very large problems. The complexity estimates for these algorithms are, however, very similar to that published in [1]. Puzzled by this discrepancy between theoretical estimates and real computational times, we have checked the implementation details of our code. To our surprise, we have realized that most of the CPU time when solving large-scale problems (as reported in [1]) was spent in a routine responsible for internal data management.

In this erratum we present computational results (CPU times) of the code after this expensive routine was carefully re-implemented. Otherwise, the details of the algorithm (including the stopping criteria and the accuracy of the results) are exactly the same as described in [1]. We only present the new results for the most relevant examples from the TOH collection (as it is called in [1])—examples with a large number of variables and a (relatively) small size of the (single) matrix constraint. The examples arise from maximum clique problems on randomly generated graphs (`theta*`) and maximum clique problems from the Second DIMACS Implementation Challenge. The dimensions of the problems are shown in Table 1.

The updated results, together with the original ones shown here for comparison, are collected in Table 2 (originally Table 9 in [1]). We have used the same computer in this table, so that the numbers are directly comparable with those in the original paper. The

Michal Kočvara: School of Mathematics, University of Birmingham, Birmingham, B15 3RU, UK, and Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 4, 18208 Praha 8, Czech Republic, e-mail: kocvara@maths.bham.ac.uk

Michael Stingl: Institute of Applied Mathematics, University of Erlangen, Martensstr. 3, 91058 Erlangen, Germany, e-mail: stingl@am.uni-erlangen.de

Table 1. (Table 7 in the original paper.) Dimensions of selected TOH problems.

problem	n	m
ham_7_5_6	1,793	128
ham_9_8	2,305	512
ham_8_3_4	16,129	256
ham_9_5_6	53,761	512
theta42	5,986	200
theta6	4,375	300
theta62	13,390	300
theta8	7,905	400
theta82	23,872	400
theta83	39,862	400
theta10	12,470	500
theta102	37,467	500
theta103	62,516	500
theta104	87,845	500
theta12	17,979	600
theta123	90,020	600
theta162	127,600	800
keller4	5,101	171
sanr200-0.7	6,033	200

slightly different iteration counts are caused by updated internal parameters in the code and by different linear algebra libraries.

To have a rough comparison with the new algorithms [2–4], we pick up the largest example `theta162`. The NCGAL code [4] solves it in 539 seconds using an Intel Xeon 3.2 GHz computer; the accuracy of the solution is comparable, though the authors get better primal feasibility. To solve the same example, the boundary point method of [3] needed 570 seconds on a Pentium 4 2.1GHz laptop.

The upgraded code PENSDP is available from the authors upon request.

References

1. M. Kočvara and M. Stingl. On the solution of large-scale sdp problems by the modified barrier method using iterative solvers. *Mathematical Programming (Series B)*, 109(2-3):413–444, 2007.
2. J. Malick, J. Povh, F. Rendl, and A. Wiegele. Regularization methods for semidefinite programming. Optimization online, http://www.optimization-online.org/db_html/2007/10/1800.html, 2007.
3. J. Povh, F. Rendl, and A. Wiegele. A boundary point method to solve semidefinite programs. Technical report, 2006.
4. X. Zhao, D. Sun, and K. C. Toh. A Newton-CG augmented Lagrangian method for semidefinite programming. Optimization online, http://www.optimization-online.org/db_html/2008/03/1930.html, 2008.

Table 2. (Table 9 in the original paper.) Results for selected TOH problems. Comparison of codes PEN-I-PCG(BFGS) (original paper and updated results) and SDPLR. CPU times in seconds; CPU/it–time per a Newton iteration; CG/it–average number of CG steps per Newton iteration. AMD Opteron 250/2.4GHz with 4GB RAM; time limit 100 000 sec.

	PEN-I-PCG(BFGS) *original paper*			PEN-I-PCG(BFGS) *updated results*			SDPLR	
problem	CPU	CPU/it	CG/it	CPU	CPU/it	CG/it	CPU	iter
ham_7_5_6	1	0.02	2	1	0.02	1	1	101
ham_9_8	33	0.77	2	36	0.73	1	13	181
ham_8_3_4	30	0.71	1	6	0.1	1	7	168
ham_9_5_6	330	7.17	1	39	0.71	1	30	90
theta42	25	0.44	9	8	0.13	8	92	6,720
theta6	24	0.44	7	25	0.33	7	257	9,781
theta62	96	1.88	10	40	0.51	10	344	6,445
theta8	93	1.55	10	51	0.73	7	395	6,946
theta82	457	7.62	14	103	1.30	12	695	6,441
theta83	1820	26.00	21	114	1.37	12	853	6,122
theta10	227	3.07	10	139	1.70	8	712	6,465
theta102	1299	16.44	13	196	2.84	14	1,231	5,857
theta103	2317	37.37	12	167	2.23	10	1,960	7,168
theta104	11953	140.62	25	283	3.37	14	2,105	6,497
theta12	254	4.62	8	299	3.88	13	1,436	7,153
theta123	10538	140.51	23	312	3.90	10	2,819	6,518
theta162	13197	173.64	13	672	8.30	10	6,004	16,845
keller4	19	0.32	9	6	0.09	7	29	2,922
sanr200-0.7	30	0.55	12	14	0.22	14	78	5,547