How Far Can We Go With Primal–Dual Interior Point Methods for SDP?

Brian Borchers
Department of Mathematics
New Mexico Tech
Socorro, NM 87801
borchers@nmt.edu

Joseph Young
Department of Mathematics
New Mexico Tech

The HKM Method

- Primal–Dual interior point methods, and in particular the HKM method, have been widely used in codes for semidefinite programming.
- CSDP, SDPA, SDPT3, and SeDuMi are all implementations of the HKM method.
- The following slides show the results of benchmarks conducted by Hans Mittelmann on a 3.2 GHz Pentium 4 with 4 gigabytes of RAM. CPU times are given in seconds. The numbers in parentheses are the number of digits of accuracy in the solution, as measured by the maximum of the DIMACS error measures.

Benchmarks

problem	PENNON	SeDuMi	SDPT3	CSDP	DSDP	SDPA
buck-3	30(1)	1414(5)	31(5)	84(6)	294(7)	24(5)
buck-4	174(2)	14544(7)	176(5)	428(7)	fail	248(6)
buck-5	2445(1)	m	2096(5)	4049(6)	fail	4960(5)
mater-3	5(5)	17(10)	25(5)	10(9)	11(8)	1044(8)
mater-4	23(4)	50(9)	204(5)	281(9)	59(8)	138452(8)
mater-5	60(4)	137(9)	988(5)	2505(9)	173(8)	m
mater-6	178	281	m	m	fail	m
shmup3	301(3)	10280(4)	307(5)	1005(8)	6559(8)	419(6)
shmup4	1655(2)	109670(8)	1712(6)	3257(7)	fail	1988(6)
shmup5	10994(2)	m	15917(5)	50575(7)	fail	m
trto-3	14(1)	1171(9)	14(4)	34(7)	12(7)	12(6)
trto-4	83(2)	11260(5)	97(3)	232(6)	96(5)	125(5)
trto-5	1149(1)	m	1076(4)	2611(5)	1516(5)	1963(4)

Benchmarks

problem	PENNON	SeDuMi	SDPT3	CSDP	DSDP	SDPA
vibra-3	27(0)	1641(8)	33(4)	69(6)	fail	28(5)
vibra-4	148(1)	16199(6)	200(4)	609(6)	fail	269(6)
vibra-5	2107(1)		2959(3)	5249(6)	>86400	4740(5)
neosfbr20	4858(2)	8569(10)	2645(9)	1608(9)	15950(9)	1663(8)
biggs	828(1)	193(9)	fail	76(9)	fail	fail
cnhil8	876(2)	103(8)	14(4)	22(8)	37(0)	31(4)
cnhil10	fail	2323(8)	146(4)	388(8)	478(0)	505(3)
cphil10	660(5)	725(13)	68(9)	91(10)	fail	337(8)
cphil12	7468(6)	m	539(10)	1052(10)	fail	3825(8)
G40_mb	915(3)	66430(8)	1806(9)	2117(8)	4835(0)	1074(4)
G40mc	997(3)	12994(9)	717(7)	443(9)	343(8)	686(9)
G48mc	fail	16398(11)	1040(9)	1006(9)	225(9)	2077(8)
G55mc	10299(2)	m	8869(7)	6532(9)	3648(8)	m
G59mc	12184(3)	m	11041(6)	8321(9)	5674(8)	m

Benchmarks

problem	PENNON	SeDuMi	SDPT3	CSDP	DSDP	SDPA
neu1	1683(1)	795(6)	fail	663(7)	fail	341(1)
neu1g	712(4)	781(9)	480(7)	829(8)	fail	279(4)
neu2	fail	1024(7)	fail	455(8)	fail	fail
neu2g	1981(4)	946(7)	428(4)	226(2)	12210(5)	fail
neu2c	2722(1)	1415(6)	fail	1748(4)	fail	1082(3)
neu3	fail	8993(7)	fail	5202(3)	fail	13938(6)
neu3g	44451(4)	12572(8)	7646(3)	9117(8)	fail	7070(6)
r1_6_0	46(1)	fail	60(7)	63(7)	93(3)	30(0)
rose13	140(3)	420(8)	78(5)	158(7)	fail	72(6)
rose15	fail	1800(6)	fail	675(4)	fail	293(4)
sdmint3	na	3764(4)	3647(4)	na	na	na
taha1a	2810(3)	838(9)	fail	1185(3)	fail	fail
taha1b	9520(4)	13646(8)	4143(7)	3192(9)	fail	10886(8)
yalsdp	1971(5)	2613(8)	855(8)	1015(9)	1487(8)	1108(8)

Benchmarks

problem	PENNON	SeDuMi	SDPT3	CSDP	DSDP	SDPA
cancer	19349(2)	m	5639	2371(7)	27140(6)	1686(4)
checker	1529(5)	m	4888(6)	8574(9)	4413(8)	m
foot	1706(1)	m	3374(4)	2861(7)	6431(1)	2156(5)
hand	278(3)	fail	496(5)	404(8)	1272(0)	253(5)
ice_2.0	9060(3)	m	m	m	52700(8)	m
p_auss2	8635	m	m	m	10090(8)	m

Benchmarks

- Overall, these benchmark results have improved considerably since the DIMACS challenge in 2000.
- These benchmark results show quite clearly that the HKM method is more robust and accurate than the augmented Lagrangian method used by PENNON and the dual interior point method used by DSDP.
- SeDuMi is the most accurate of the HKM codes, but is generally slower than the other codes.
- Most of these problems can be solved in two hours or less.
- However, the HKM based codes cannot solve some of the larger problems because they run out of memory.

Computational Complexity

- Multiplying matrices of size n takes $O(n^3)$ time.
- Factoring matrices of size n takes $O(n^3)$ time.
- For dense constraint matrices, constructing O takes $O(mn^3 + m^2n^2)$ time.
- For sparse constraint matrices with $O(1)$ entries, constructing the Schur complement matrix takes $O(mn^2 + m^2)$ time.
- Overall, for problems with sparse constraint matrices, iterations of the HKM method take $O(m^3 + n^3)$ time.

Storage Requirements

- Consider an SDP problem with m constraints, and block diagonal matrix variables X and Z with blocks of size n_1, n_2, ..., n_k.
- The algorithm requires storage for an m by m Schur complement matrix. This matrix is (with very few exceptions) fully dense.
- The algorithm requires storage for several block diagonal matrices with blocks of size n_1, n_2, ..., n_k.
- Blocks of X and related matrices are typically fully dense, while blocks of Z and related matrices may be sparse.

Storage Requirements

- In practice, the constraint matrices A_1, A_2, ..., A_m are typically quite sparse.
- Assuming that the storage required for each constraint matrix A_i is $O(1)$, the storage required by the HKM method is $O(m^2 + n^2)$.
- For example, the storage required by CSDP is approximately $8(m^2 + 13(n_1^2 + n_2^2 + \ldots + n_k^2))$ bytes.

Some Particular Problems.

Problem	m	n	Storage
trto-5	3280	1761	0.4 GB
buck-5	3280	1760; 1761	0.8 GB
vibra-5	3280	1760; 1761	0.8 GB
checker	3970	3970	2 GB
G59mc	5000	5000	3 GB
torusg3-15	3375	3375	3 GB
toruspm3-15-50	3375	3375	3 GB
hamming_8_3_4	16129	256	3 GB
mater-6	20463	4968x11	5 GB
ice_2.0	8113	8113	10 GB
p_auss2_3.0	9115	9115	12 GB

Looking Ahead to 2010

- In 1965, Intel's Gordon Moore pointed out that the number of transistors on an integrated circuit was doubling about every 18 months. For the last forty years, this exponential growth has continued.
- This trend has also resulted in an approximate doubling of the capacity of RAM memory and a doubling of the speed of microprocessors every 18 months.
- My current desktop computer has a 3.0 GHz Pentium 4 processor and 2 gigabytes of RAM.
- If Moore's law holds, then I expect that if I buy a new machine in 2010, it will have 16 to 32 gigabytes of RAM, and a processor that is about 16 times faster than my 3.0 GHz Pentium 4.

64 Bit Computing

- The current generation of Intel Pentium 4 and AMD Athlon processors are 32 bit processors. This means that programs can directly address only up to 2^{32} bytes of storage (4 gigabytes.)
- In practice, operating system software further limits this to 2 gigabytes for Windows, and 3 gigabytes for Linux.
- AMD's new Opteron processors, as well as G5 processors in Macintosh computers are 64 bit processors. This means that programs on these processors can address far more than 4 gigabytes of storage.
- Intel has announced that their next generation of Pentium processors will be 64 bit processors.
- Most workstation vendors (HP, IBM, SGI, etc.) have long had 64 bit processors.

64 Bit Computing

- The programming model for 64 bit processors is commonly I32LP64. That is, integers are stored as 32 bit numbers, while long integers and pointers are 64 bits.
- Well written codes should work on 64 bit processors with very few changes. In practice, many codes have to be "cleaned up" to run in 64 bit mode.
- The current version of MATLAB does not support 64 bit processing. However, MATLAB 7 (to be released this summer) is expected to do so.
- Another important issue is the availability of 64 bit versions of the BLAS and LAPACK libraries.

Multiprocessing

- Many manufacturers are already producing multiprocessor systems.
- As the density of transistors on integrated circuits has increased, manufacturers have found it increasing difficult to make effective use of the available transistors.
- One simple approach is to put multiple (typically 2 or 4) microprocessors on a single chip.
- Intel, AMD, and Sun have all announced plans to produce such chips in the next few years.
- This means that we will have to develop parallel algorithms for SDP to make effective use of these processors.

A 64-bit Parallel Version of CSDP

- We ported CSDP 4.7 to an IBM p690 computer at the National Center for Supercomputer Applications (NCSA).
- The p690 is a cluster of nodes. Each node is itself a shared memory multiprocessor.
- NCSA has several nodes, with each node having up to 32 processors and 256 gigabytes of RAM.
- The p690 uses 1.3 GHz Power 4 processors.
- IBM's Extended Scientific Subroutine Library (ESSL) provides parallel versions of the BLAS and LAPACK routines used by CSDP.

A 64-bit Parallel Version of CSDP

- For our initial experiments, we simply compiled CSDP 4.7 and linked it with the multiprocessor version of ESSL.
- No changes were made to the source code!
- Using four processors, the code typically runs about twice as fast as on a single processor.
- Although the BLAS/LAPACK operations are automatically parallelized, building the Schur complement matrix can become a scalar bottleneck on some problems.
- Efforts to improve the parallel speedup are ongoing.

Benchmarks

Problem	m	n	O	Factor	Other	1	2	4
hamming98	2305	512	5%	75%	20%	0:59	0:38	0:27
copo14	1275	560	3%	90%	7%	0:07	0:04	0:03
torusg3-8	512	512	1%	6%	93%	0:23	0:16	0:11
control11	1596	165	79%	20%	1%	4:30	4:04	3:43
theta6	4375	300	2%	95%	3%	4:04	2:23	1:45
equalG11	801	801	12%	4%	84%	2:13	1:26	0:58
maxG11	800	800	<1%	7%	93%	0:38	0:27	0:21
qpG11	800	1600	<1%	1%	98%	4:50	3:16	2:24

Benchmarks

Problem	m	n	Memory	Error	Time
torusg3-15	3375	3375	3GB	1.3e-09	22:29
toruspm3-15-50	3375	3375	3GB	5.4e-09	22:26
hamming_8_3_4	16129	256	3GB	2.0e-09	54:42
mater-6	20463	4968x11	5GB	2.4e-09	3:07:05
ice_2.0	8113	8113	10GB	9.9e-09	15:21:38
p_auss2_3.0	9115	9115	12GB	7.1e-09	13:51:16

Getting CSDP

The current version of CSDP is version 4.7. The package includes a stand–alone solver, a subroutine interface to the solver, and a MATLAB interface to the solver. CSDP is an open source project, available under the Common Public License (CPL). You can download the source code, some binary versions, and a user's guide from

`http://www.nmt.edu/~borchers/csdp.html`

Hans Mittelmann's benchmarks can be found at

`ftp://plato.la.asu.edu/pub/sparse_sdp.txt`