CoCoA-5 - Slug #709

Speed of some Normaliz calls (NmzComputation)

17 May 2015 10:17 - John Abbott

Status:	In Progress	**Start date:**	17 May 2015
Priority:	High	**Due date:**	
Assignee:		**% Done:**	10%
Category:	External Libs	**Estimated time:**	0.00 hour
Target version:	CoCoA-5.4.2	**Spent time:**	2.35 hours

Description

JAA has done a few speed comparisons between calling normaliz directly from the command line and calling NmzComputation.

Sometimes the computation times are very similar; sometimes NmzComputation can be noticably slower. A particular example is 5x5.in:

- normaliz takes about 9mins on my machine
- NmzComputation takes about 14mins on my machine

Is this difference just "data transmission"? Perhaps memory management?

History

#1 - 17 May 2015 10:33 - John Abbott

TO DO JAA can try the same computation but from CoCoALib; perhaps some time is spent converting to CoCoA-5 data structures?

#2 - 21 May 2015 11:15 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here is the input I was giving to CoCoA-5

```
// taken from 5x5.in
// directly with Normaliz takes about 9min; via CoCoA takes about 14mins WHY???
M := mat([
[1, 1, 1, 1, 1, -1, -1, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
[1, 1, 1, 1, 1,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
[1, 1, 1, 1, 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1,  0,  0,  0,  0],
[1, 1, 1, 1, 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1],
[0, 1, 1, 1, 1, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0],
[1, 0, 1, 1, 1,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0],
[1, 1, 0, 1, 1,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0],
[1, 1, 1, 0, 1,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0],
[1, 1, 1, 1, 0,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1],
[0, 1, 1, 1, 1,  0, -1,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0, -1],
      [1, 1, 1, 1, 0,  0,  0,  0, -1,  0,  0,  0, -1,  0,  0,  0, -1,  0,  0,  0, -1,  0,  0,  0,  0]]);

G := mat([[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]);

cone := record[ equations := M, grading := G];
t0 := CpuTime();
nmz := NmzComputation(cone);
println "NMZ time: ", TimeFrom(t0);
// indent(ans); -- rather big!!!!
```

#3 - 21 May 2015 11:17 - John Abbott

Here is body of the CoCoALib code I used to speed testing in CoCoALib:

```
void program()
{
  GlobalManager CoCoAFoundations;

#ifndef CoCoA_WITH_NORMALIZ
  cout << "Normaliz library is not available to CoCoALib." << endl;
#else // NORMALIZ is available
  cout << boolalpha; // so that bools print out as true/false

  int nrows, ncols;
  cin >> nrows;
  cin >> ncols;

  vector<vector<BigInt> > M(nrows, vector<BigInt>(ncols));
  for (int i=0; i < nrows; ++i)
    for (int j=0; j < ncols; ++j)
    {
      cin >> M[i][j];
    }

  cin >> nrows;
  cin >> ncols;

  vector<vector<BigInt> > G(nrows, vector<BigInt>(ncols));
  for (int i=0; i < nrows; ++i)
    for (int j=0; j < ncols; ++j)
    {
      cin >> G[i][j];
    }

  libnormaliz::Type::InputType InputType = libnormaliz::Type::equations;

  std::map< libnormaliz::InputType, std::vector<std::vector<BigInt> > > CtorArgs;
  CtorArgs[libnormaliz::Type::equations] = M;
  CtorArgs[libnormaliz::Type::grading] = G;

  Normaliz::cone C3(CtorArgs);
  cout << "Cone created: " << endl << C3 << endl << endl;
  vector<vector<BigInt> > res;
  double t0;
  t0 = CpuTime();
  C3.myComputation(); // compute everything
  cout << "time everything  -> " << CpuTime()-t0 << endl;

#endif // CoCoA_WITH_NORMALIZ
}
```

#4 - 21 May 2015 11:19 - John Abbott

Here is the original Normaliz input file (called 5x5.in in their source tree):

```
11
25
1  1  1  1  1 -1 -1 -1 -1 -1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1  1  1  1  1   0   0   0   0   0 -1 -1 -1 -1 -1   0   0   0   0   0   0   0   0   0
1  1  1  1  1   0   0   0   0   0   0   0   0   0   0 -1 -1 -1 -1 -1   0   0   0   0   0
1  1  1  1  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 -1 -1 -1 -1 -1
0  1  1  1  1 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0
1  0  1  1  1   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0
1  1  0  1  1   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0
1  1  1  0  1   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0
1  1  1  1  0   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1   0   0   0   0 -1
0  1  1  1  1   0 -1   0   0   0   0   0 -1   0   0   0   0   0 -1   0   0   0   0   0 -1
1  1  1  1  0   0   0   0 -1   0   0   0 -1   0   0   0 -1   0   0   0 -1   0   0   0   0
equations
1
25
1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
grading
```

IMPORTANT NOTE to use this as input for the C++ program above, **delete the word equations**

#5 - 21 May 2015 11:26 - John Abbott

On my machine (MacBook Pro with OSX 10.5.8, compiler version 4.2.1)

Normaliz 2.12 took 9mins (about 550s)
CoCoA-5 took 14mins (about 840s)
CoCoALib took 13mins (about 790s)

#6 - 21 May 2015 11:34 - John Abbott

With Christof we ran a side-by-side comparison (on my machine) with "verbose" activated.
The run under CoCoA-5 was apparently uniformly slower than the run directly in Normaliz. In particular we excluded the possibility that the extra time was copying the answer from Normaliz data structures to CoCoA ones.

NOTE: Christof also noted curious behaviour running Normaliz inside GAP; the process seemed to go into swap for no good reason. He is investigating. we have no idea whether this may be related to the strange behaviour seen with CoCoA/CoCoALib.

#7 - 21 May 2015 11:57 - John Abbott

Attempts to profile the running program on my computer failed :-(
gprof does not work properly in MacOSX, and there were problems in the Linux VM too.

Currently stumped.

#8 - 17 Feb 2016 13:08 - John Abbott

- Priority changed from Normal to High

- Target version changed from CoCoA-5.1.3/4 Jan 2016 to CoCoA-5.2.0 spring 2017

Postponing this to next version. Now I have access to a proper linux box, maybe I can investigate properly this mysterious slug... I can hope it was a MacOS weirdness ;-)

Pushing priority up to "high"; it would definitely be good to understand what is going on here.

#9 - 17 Feb 2016 13:42 - Christof Soeger

For the problem I discovered in the NormalizInterface for GAP we found the problem meanwhile. GAP used to set its own (really stupid) memory manager for the gmpxx classes which lead to a significant slowdown if they were used. See
https://github.com/gap-packages/NormalizInterface/issues/17

You could check if it also happens if you call Normaliz::HilbertBasis(C3); instead of C3.myComputation();, there should be almost no use of gmp. (And it should be considerably faster).

#10 - 14 Oct 2016 16:34 - John Abbott

I have reconfirmed that there is strange behaviour on the Linux machine here in Kassel. Compiler is g++ version 5.3.1.

If I perform the computation in CoCoA-5, the time taken is about 120s (whether or not I specify UseGMPAllocator).

If I do the computation using the C++ code above, the time taken is about 225s (with UseGMPAllocator being marginally slower).

Why should CoCoALib be so much slower? Maybe I'd better check the inputs!

I was unable to compile Normaliz (version 2.99.4) because linking failed: it did not find

```
/usr/bin/ld: cannot find -lstdc++
/usr/bin/ld: cannot find -lm
/usr/bin/ld: cannot find -lc
```

No idea why there should be a problem :-(

#11 - 27 Apr 2017 15:30 - John Abbott

- Target version changed from CoCoA-5.2.0 spring 2017 to CoCoA-5.2.2

Postponing as this is likely to take some time to comprehend and resolve. It is also "not critical" in that there is no risk of a wrong answer (you may simply have to wait longer to get the answer...).

#12 - 13 Nov 2017 16:58 - John Abbott

- Target version changed from CoCoA-5.2.2 to CoCoA-5.2.4

#13 - 26 Jul 2018 11:08 - John Abbott

- Target version changed from CoCoA-5.2.4 to CoCoA-5.3.0

#14 - 01 Oct 2019 14:31 - John Abbott

- Target version changed from CoCoA-5.3.0 to CoCoA-5.4.0

We should check what happens with current (2019-10-01) versions of CoCoALib and Normaliz.

#15 - 12 Oct 2021 09:33 - John Abbott

(2021-10-12) Last time this was checked was 4 years ago! We must check to see if it still a problem!

#16 - 03 Feb 2022 19:23 - John Abbott

- Target version changed from CoCoA-5.4.0 to CoCoA-5.4.2