Amsterdam Optimization Modeling Group LLC

Notes on the IFPRI Spatial Production Allocation NLP Model

This document describes some experiments with the Spatial Production Allocation NLP model LANDALOC_feas.gms.

THIS WORK WAS PERFORMED UNDER A CONTRACT BY GAMS DEVELOPMENT CORP., WASHINGTON DC.

Erwin
11/14/2008

INTRODUCTION

The model LANDALOC_feas.gms was provided to me in order to see if this model could be solved efficiently on a Linux cluster using parallel processing techniques.

LINUX

GAMS and most of its solvers are available both on Windows and Linux. In general performance should be similar. The Linux version has no IDE (Integrated Development Environment) and no Excel interface.

GAMS models are in general portable from Windows to Linux. I.e. a model developed on Windows should run without change on a Linux box. Sometimes the names of include files need to be updated: windows uses a different way to write directories (e.g. '\' vs '/') and the Linux file system is case sensitive while Windows' filenames are not.

PARALLEL PROCESSING

GAMS by itself is not parallel but some of the solvers are available in parallel versions:

- Cplex: LP, MIP, QCP
- Xpress: LP, MIP, QCP
- Mosek: LP, MIP, QCP, NLP (convex only)

These solvers use a Shared Memory model. There is no off-the-shelf solver available supporting a Distributed Memory model. This means that parallel processing is restricted to multi-cpu and multi-core machines. It is not possible to distribute a problem over different machines.

GAMS has some rudimentary "grid-computing" facilities, but that is aimed at solving many problems in parallel instead of running one big model.

Although parallel processing for MIP models can work very well (with excellent speed-ups), parallel processing for LP and NLP is largely restricted to barrier type (interior point) algorithms with modest speed-ups.

LP MODEL

The first part is a large LP. The statistics are:

LP model	
name:	Landfeas
rows:	35734
cols:	550336
nz:	1701683

This model can be solved quickly by using a state of the art LP solver. Here are some results:

solver	Time	obj	notes
conopt	2495.532	11.6806	conopt is an NLP solver, and not very fast for LP's
minos	919.891	11.6806	EXIT - The objective has not changed in many iterations.
coincbc	86.485	11.6806	default settings (1 thread)
cplex	13.926	11.6806	default settings (this is the winner)
cplex	54.360	11.6806	barrier (4 threads) slower than default (dual simplex)
mosek	74.389	11.6809	default settings (1 thread)
mosek	71.011	11.6806	parallel version (4 threads) just slightly faster
mosek	22.609	11.6806	dual simplex (1 thread)
xpress	32.630	11.6849	default settings (1 thread)
xpress	80.033	11.6806	barrier 4 threads

The commercial high-end solvers do a good job on this model. Parallel processing has limited value as in many cases the default settings using 1 thread outperform other settings.

It is noted that for LP's only the barrier (interior point) method is parallelized. On this model the simplex method (serial) is faster.

NLP MODEL

This is a very large NLP. The statistics are:

NLP Model:	
name:	LandentSN
rows:	28792
cols:	546866
nz:	2771109
nlnz:	543391

The NLP solver that was used by IFPRI was Conopt and an option file with

rtredg = 1.0e-4

was used. This may cause early termination, before the optimum has been reached. We believe this is actually what is happening in our case. We tried a few other NLP solvers but none of the solvers could find the optimum solution quickly. The best result was obtained by Mosek, which reached a "near optimal" solution quickly and then terminated.

solver	time	obj	Notes
conopt	9455.158	202.0981	Uses rtredg = 1.0e-4, terminates maybe too early
mosek	582.648	101.7951	4 threads, status=near optimal

It is noted that we reformulated the model slightly to be able to run with Mosek. We formed a nonlinear objective with linear constraints:

```
equation rdef3;
rdef3.. ENTROPY =E= (SUM((i,j)$NONZERO(i,j),ALLOC(i,j)*(LOG([ALLOC(i,j) +
epsilon]/[REV(i,j) + epsilon])))/SCALELP ;

MODEL LANDentSN2 /LANDTOT, SUBCROP, SUMONE, IRRLIMIT, RDEF3/;

option nlp=mosek;
LANDENTSN2.optfile=1;
SOLVE LANDENTSN2 USING NLP MINIMIZING ENTROPY;
```

DERIVATIVE CALCULATION

It is important to pay attention to the form of the objective. A smaller test model will illustrate this.

```
  Sets
       i   canning plants   / seattle, san-diego /
       j   markets          / new-york, chicago, topeka /
       k   multiplicity     / 1*10000/
;

  Parameters

       a(i)  capacity of plant i in cases
         /    seattle     350
              san-diego   600  /

       b(j)  demand at market j in cases
         /    new-york    325
              chicago     300
              topeka      275  / ;

  Table d(i,j)  distance in thousands of miles
                    new-york       chicago      topeka
      seattle          2.5           1.7          1.8
      san-diego        2.5           1.8          1.4  ;

  Scalar f  freight in dollars per case per thousand miles  /90/ ;

  Parameter c(i,j)  transport cost in thousands of dollars per case ;

            c(i,j) = f * d(i,j) / 1000 ;

  Variables
       x(i,j,k)  shipment quantities in cases
       z       total transportation costs in thousands of dollars ;

  Positive Variable x ;

  Equations
       cost1        define objective function
       cost2        define objective function
       supply(i,k)   observe supply limit at plant i
```

```
       demand(j,k)    satisfy demand at market j ;

  scalar e /0.000001/;

* IFPRI VERSION    NL + LINEAR
  cost1 ..         z  =e=  sum((i,j,k), x(i,j,k)*log(x(i,j,k)+e)) -
                           sum((i,j,k), x(i,j,k)*log(c(i,j)+e));

* EK VERSION       NL ONLY
  cost2 ..         z  =e=  sum((i,j,k),
                           x(i,j,k)*log((x(i,j,k)+e)/(c(i,j)+e)));

  supply(i,k) ..    sum(j, x(i,j,k))  =l=  a(i) ;

  demand(j,k) ..    sum(i, x(i,j,k))  =g=  b(j) ;

option limrow=0;
option limcol=0;

  Model t1 /cost1,supply,demand/ ;
  Model t2 /cost2,supply,demand/ ;

  Solve t1 using nlp minimizing z ;
*  Solve t2 using nlp minimizing z ;
```

This is essentially the transport model where we add an extra index to be able to make the model bigger, and where we use an entropy objective. Model t2 solves much faster than model t1. The solver IPOPT demonstrates this nicely:

Model t1	Total CPU secs in IPOPT (w/o function evaluations) = 6.033 Total CPU secs in NLP function evaluations = 33.971
Model t2	Total CPU secs in IPOPT (w/o function evaluations) = 5.905 Total CPU secs in NLP function evaluations = 0.541

This shows us that t1 and t2 are very much identical to the solver, but they differ very much with respect to function- and derivative evaluation. I suspect that the linear part in objective cost1 is stored separately with an inadequate fast lookup.

For smaller models the derivative calculation does not need attention, but for a model of this size, it is very important to make sure function and derivative evaluation is a small fraction of the total solution time.

CONVEXITY

The model is suited for Mosek as the objective is convex. This can be seen by inspecting the function

$$f(x) = x \log(x + a)$$

The second derivatives are:

$$\frac{d^2 f}{dx^2} = \frac{2}{x+a} - \frac{x}{(x+a)^2} = \frac{x+2a}{(x+a)^2} > 0$$

for any positive x. This confirms the problem is convex and Mosek is valid choice for solving this problem.

IMPROVEMENT BY SUCCESSIVE QP'S

In order to further improve the solution we tried to restart from the Mosek solution. This did not give good results. Conopt declared the model infeasible, and other solvers did not make much progress.

In the end we formed a QP (Quadratic Programming Problem) by taking the 2nd order Taylor series approximation of x*log(x+e). We then solved a series of QP's where we replace f(x) by

$$f(x) \approx f(x^0) + \nabla f(x^0)\,(x - x^0) + \frac{\nabla^2 f(x^0)}{2}(x - x^0)^2$$

We stop when the objective does not change anymore.

Mosek only solves convex QP's, which in this case is not a problem as the QP's are all convex.

In GAMS notation:

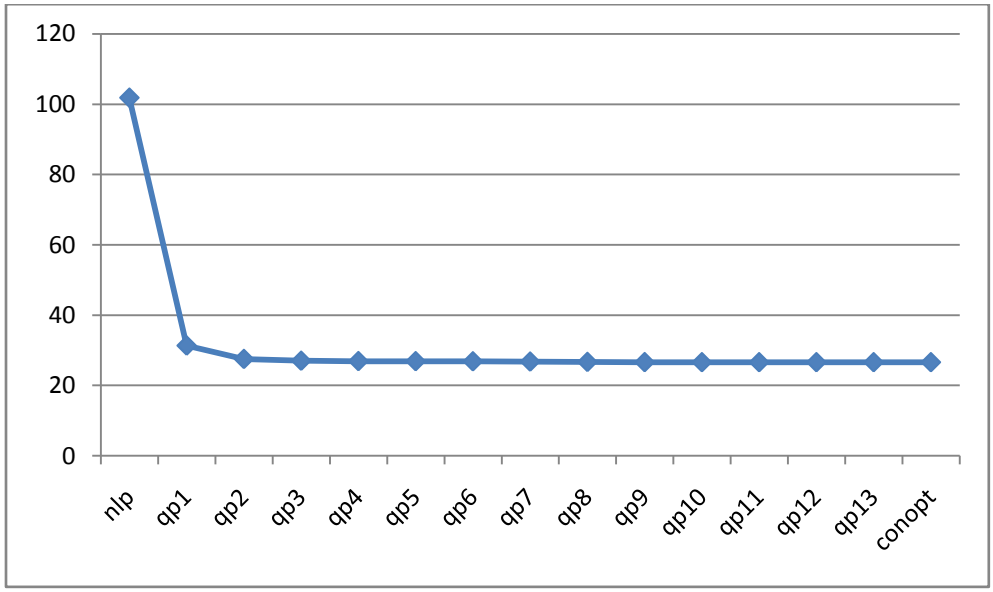
```
equation rdef5 "2nd order taylor approx";

rdef5.. ENTROPY =E= [SUM(NONZERO(i,j),
*      f =  ALLOC.l(i,j)*LOG(ALLOC.l(i,j) + epsilon)
*      f' =  ALLOC.l(i,j)/(ALLOC.l(i,j) + epsilon) + log(ALLOC.l(i,j) + epsilon)
*      f'' = 2/(ALLOC.l(i,j) + epsilon) - ALLOC.l(i,j)/sqr(ALLOC.l(i,j) + epsilon)
                      ALLOC.l(i,j)*LOG(ALLOC.l(i,j) + epsilon)
                      +
                      [ALLOC.l(i,j)/(ALLOC.l(i,j) + epsilon)  +  log(ALLOC.l(i,j) +
epsilon)] * (ALLOC(i,j)-ALLOC.L(i,j))
                      +
                      0.5*[2/(ALLOC.l(i,j) + epsilon)  -
ALLOC.l(i,j)/sqr(ALLOC.l(i,j) + epsilon)] * sqr(ALLOC(i,j)-ALLOC.L(i,j))

                          )
                    -
                    SUM(NONZERO(i,j),ALLOC(i,j)*LOG(REV(i,j) + epsilon))] /SCALELP;



MODEL mapprox /LANDTOT, SUBCROP, SUMONE, IRRLIMIT, RDEF5/;
option qcp=mosek;

set iter /1*100/;
parameter objApprox(iter);
scalar done /0/;
loop(iter$(not done),
   solve mapprox using qcp minimizing entropy;
   objApprox(iter) = entropy.l;
   display objApprox;
   done$(abs(objApprox(iter)-objApprox(iter-1)) <= 1.0e-5) = 1;
);
```

We used parallel Mosek (4 threads) to solve the QP's. Finally we asked Conopt to improve the solution further. The results are:

model	time	obj
nlp	582.648	101.7951
qp1	73.003	31.3121

qp2	70.156	27.508
qp3	65.358	27.0025
qp4	64.346	26.8975
qp5	67.953	26.8712
qp6	69.314	26.8353
qp7	70.216	26.7588
qp8	72.137	26.6631
qp9	72.846	26.6035
qp10	94.495	26.5873
qp11	98.648	26.5851
qp12	92.613	26.585
qp13	93.819	26.585
conopt	538.999	26.585

If we were less conservative we could have terminated a little bit earlier, as the graph of the objective values illustrates.



MOSEK CAVEATS

In the above I have used Mosek as it outperformed other solvers for this model. There are however a few quality issues with this solver itself and the GAMS link. Before purchasing Mosek you may want to be aware of these issues. Here is a short list of known problems:

CONVEXITY

Mosek can only handle convex problems. It may or may not give a reasonable error message if your model is not convex. Sometimes the message is very misleading or incomprehensible. The developers claim it is the modelers task to make sure the model is convex.

MOSEK MIQCP ISSUE

This is a trivial MIQCP:

```
variable z;
binary variable x;
equation e;
e.. z =e= sqr(x);
model m/e/;
solve m minimizing z using miqcp;
```

Embarrassingly, Mosek cannot solve this. The log file mentions

Return code - 1050 [MSK_RES_ERR_UNKNOWN]

The GAMS link is not perfect either. It shows:

**** SOLVER STATUS 1 NORMAL COMPLETION
**** MODEL STATUS 14 NO SOLUTION RETURNED
**** OBJECTIVE VALUE 0.0000

This is a strange combination and no further explanation is given.

GAMS/MOSEK ERROR REPORTING 1

The listing file does not give any explanation in case something goes wrong. The previous section has an example. Another example is how the NLP solution is presented:

```
               S O L V E      S U M M A R Y

     MODEL   LANDentSN2          OBJECTIVE  ENTROPY
     TYPE    NLP                 DIRECTION  MINIMIZE
     SOLVER  MOSEK               FROM LINE  249751

**** SOLVER STATUS     4 TERMINATED BY SOLVER
**** MODEL STATUS      7 INTERMEDIATE NONOPTIMAL
**** OBJECTIVE VALUE              101.7951

 RESOURCE USAGE, LIMIT        593.428    900000.000
 ITERATION COUNT, LIMIT         0        900000
 EVALUATION ERRORS              0             0

 MOSEK Link       Aug  1, 2008 22.8.1 WEX 5438.6015 WEI x86_64/MS Windows

 M O S E K       version 5.0.0.90 (Build date: Jun  6 2008 14:57:22)
 Copyright (C)   MOSEK ApS, Fruebjergvej 3, Box 16
                 DK-2100 Copenhagen, Denmark
                 http://www.mosek.com
```

Again: no explanation is provided in the listing file why Mosek stopped. Sometimes you can decipher something from the log file (unfortunately on Linux this is the screen, so it may have disappeared).

GAMS/MOSEK ERROR REPORTING 2

The log file presents cryptic error messages like:

```
Return code - 1500 [MSK_RES_ERR_INV_PROBLEM].
```

One would expect such codes to be translated in somewhat readable English. Currently you have to search the manual for a short explanation of these error codes.

GAMS/MOSEK ERROR REPORTING 3

In some cases unusual or contradictory error returns are presented. Here is an example:

```
**** SOLVER STATUS 8 USER INTERRUPT
**** MODEL STATUS 1 OPTIMAL
```

MOSEK INFEASIBILITY REPORTING

Mosek reports infeasibilities in a non-standard way. See the documentation for some notes.

MOSEK DOCUMENTATION

The documentation is not always clear. For instance the first parameter discussed in section 5 may lead you to believe it can solve non-convex problems using `MSK_IPAR_OPTIMIZER=5` which is not the case.

CONCLUSION

The LP part of the model can be solved with any good LP solver very efficiently.

The NLP part of the model is very complicated. It is both very large and numerically challenging. The most efficient approach was using Mosek to solve the NLP and then use a series of QP's to improve the solution. The total solution time is 30-40 minutes.