CONVERT

Contents

1 Introduction

CONVERT is a utility which transforms a GAMS model instance into a scalar model where all confidential information has been removed or into formats used by other modeling and solution systems. CONVERT is designed to achieve the following goals:

- Permit users to convert a confidential model into GAMS scalar format so that any idenifiable structure is removed. It can then be passed on to others for investigation without confidentiality being lost.
- A way of sharing GAMS test problems for use with other modeling systems or solvers.

CONVERT comes free of charge with any licensed GAMS system and can convert GAMS models into the following formats:

- AlphaECP
- AMPL
- AmplNLC
- BARON
- CplexLP
- CplexMPS
- Dict
- FixedMPS
- GAMS (Scalar format)
- Jacobian
- LAGO
- LGO
- LindoMPI

- LINGO
- LocalSolver
- MINOPT
- NLP2MCP
- ViennaDag

For more information see the options section.

2 How to use CONVERT

CONVERT is run like any other GAMS solver. From the command line this is:

```
>> gams modelname modeltype=convert
```

where `modelname` is the GAMS model name and `modeltype` the solver indicator for a particular model type (e.g. LP, MIP, RMIP, QCP, MIQCP, RMIQCP, NLP, DNLP, CNS, MINLP, or MCP). CONVERT can also be specified via the option statement within the model itself before the solve statement:

```
option modeltype=convert;
```

3 The GAMS Scalar Format

By default, CONVERT generates a scalar GAMS model (`gams.gms`) from the input model. The scalar model exhibits the following characteristics:

- A model without sets or indexed parameters. It does not exhibit any of the advanced characteristics of modeling systems and is easily transformable.
- A model with a new set of individual variables, depicting each variable in the GAMS model as one of 3 types: positive, integer or binary. Each variable is numbered sequentially, i.e. all positive GAMS variables are mapped into n single variables `x1, x2, ..., xn`.
- A model with individual equations depicting each variable in the GAMS model. All equations are also numbered sequentially, that is equations `e1, e2, ..., em`.

Equation and variable bounds, as well as variable starting values are preserved from the original GAMS formulation.

As an example, suppose the user wishes to translate the GAMS Model Library model `trnsport.gms` into scalar format, One would run `gams trnsport.gms lp=convert`, which would generate the following scalar model `gams.gms`:

```
*  LP written by GAMS Convert at 07/29/04 12:59:58
*
*  Equation counts
*     Total       E       G       L       N       X       C
*         6       1       3       2       0       0       0
*
*  Variable counts
*                 x       b       i     s1s     s2s      sc      si
*     Total    cont  binary integer    sos1    sos2   scont    sint
*         7       7       0       0       0       0       0       0
*  FX     0       0       0       0       0       0       0       0
```

```
*
*  Nonzero counts
*     Total   const      NL     DLL
*        19      19       0       0
*
*  Solve m using LP minimizing x7;

Variables  x1,x2,x3,x4,x5,x6,x7;
Positive Variables  x1,x2,x3,x4,x5,x6;
Equations  e1,e2,e3,e4,e5,e6;

e1..  - 0.225*x1 - 0.153*x2 - 0.162*x3 - 0.225*x4 - 0.162*x5 - 0.126*x6 + x7
      =E= 0;
e2..    x1 + x2 + x3 =L= 350;
e3..    x4 + x5 + x6 =L= 600;
e4..    x1 + x4 =G= 325;
e5..    x2 + x5 =G= 300;
e6..    x3 + x6 =G= 275;

* set non default bounds

* set non default levels

* set non default marginals

Model m / all /;
m.limrow=0; m.limcol=0;

Solve m using LP minimizing x7;
```

Note that the resulting scalar model does not contain any of the descriptive information about the data or the context of the constraints.

Additionally, a dictionary file (`dict.txt`) is created by default which specifies a mapping between the variable and equation names in the scalar model and their corresponding names in the original model.

For the above example, the dictionary file is

```
LP written by GAMS Convert at 07/29/04 12:59:59

Equation counts
    Total        E        G        L        N        X        C        B
        6        1        3        2        0        0        0        0

Variable counts
                 x        b        i      s1s      s2s       sc       si
    Total     cont   binary  integer     sos1     sos2    scont     sint
        7        7        0        0        0        0        0        0
FX      0        0        0        0        0        0        0        0

Nonzero counts
    Total    const       NL      DLL
       19       19        0        0

Equations 1 to 6
  e1  cost
  e2  supply(seattle)
  e3  supply(san-diego)
```

```
  e4  demand(new-york)
  e5  demand(chicago)
  e6  demand(topeka)

Variables 1 to 7
  x1  x(seattle,new-york)
  x2  x(seattle,chicago)
  x3  x(seattle,topeka)
  x4  x(san-diego,new-york)
  x5  x(san-diego,chicago)
  x6  x(san-diego,topeka)
  x7  z
```

Conversion of a GAMS model to a scalar one may be handy for model debugging. However, in this case, it may be good to retain the original variable and equation names. The following simple sed command attempts to achieve this:

```
sed ‘sed -n -e "s:^  \([exbi][0-9]*\)  \(.*\):-e s/\1/\2/g:gp" dict.txt \
 | sed -n ’1!G;h;$p’‘ gams.gms
```

For the above example, this outputs:

```
Variables x(seattle,new-york),x(seattle,chicago),x(seattle,topeka),x(san-diego,new-york),
 x(san-diego,chicago),x(san-diego,topeka),z;

Positive Variables x(seattle,new-york),x(seattle,chicago),x(seattle,topeka),
 x(san-diego,new-york),x(san-diego,chicago),x(san-diego,topeka);

Equations cost,supply(seattle),supply(san-diego),demand(new-york),demand(chicago),demand(topeka);


cost..  - 0.225*x(seattle,new-york) - 0.153*x(seattle,chicago) - 0.162*x(seattle,topeka)
 - 0.225*x(san-diego,new-york) - 0.162*x(san-diego,chicago) - 0.126*x(san-diego,topeka) + z
      =E= 0;

supply(seattle)..    x(seattle,new-york) + x(seattle,chicago) + x(seattle,topeka) =L= 350;

supply(san-diego)..    x(san-diego,new-york) + x(san-diego,chicago) + x(san-diego,topeka) =L= 600;

demand(new-york)..    x(seattle,new-york) + x(san-diego,new-york) =G= 325;

demand(chicago)..    x(seattle,chicago) + x(san-diego,chicago) =G= 300;

demand(topeka)..    x(seattle,topeka) + x(san-diego,topeka) =G= 275;
```

Of course, this is not a valid GAMS code and cannot be compiled, but it may be sufficient to see the model algebra as generated by the GAMS compiler.

By using

```
sed ‘sed -n -e "y/(),-/____/" -e "s:^  \([exbi][0-9]*\)  \(.*\):-e s/\1/\2/g:gp" dict.txt \
  | sed -n ’1!G;h;$p’‘ gams.gms
```

one gets for this example

```
Variables  x_seattle_new_york_,x_seattle_chicago_,x_seattle_topeka_,x_san_diego_new_york_,
 x_san_diego_chicago_,x_san_diego_topeka_,z;
```

```
Positive Variables  x_seattle_new_york_,x_seattle_chicago_,x_seattle_topeka_,x_san_diego_new_york_,
 x_san_diego_chicago_,x_san_diego_topeka_;

Equations  cost,supply_seattle_,supply_san_diego_,demand_new_york_,demand_chicago_,demand_topeka_;


cost..  - 0.225*x_seattle_new_york_ - 0.153*x_seattle_chicago_ - 0.162*x_seattle_topeka_
  - 0.225*x_san_diego_new_york_ - 0.162*x_san_diego_chicago_ - 0.126*x_san_diego_topeka_ + z
      =E= 0;

supply_seattle_..    x_seattle_new_york_ + x_seattle_chicago_ + x_seattle_topeka_ =L= 350;

supply_san_diego_..    x_san_diego_new_york_ + x_san_diego_chicago_ + x_san_diego_topeka_ =L= 600;

demand_new_york_..    x_seattle_new_york_ + x_san_diego_new_york_ =G= 325;

demand_chicago_..    x_seattle_chicago_ + x_san_diego_chicago_ =G= 300;

demand_topeka_..    x_seattle_topeka_ + x_san_diego_topeka_ =G= 275;
```

This can even be compiled by GAMS and gives the correct solution.

The proposed commands come with several limitations and may not produce in all cases the desired output. For example, wrong results would be printed if the original model contains variable or equation names that start with {b,i,e,x}[digit].

Also semicontinuous or semiinteger variables or special ordered sets are not supported by the above. We will leave it to the experienced user to extend the command appropriately.

4 User-Specified Options

CONVERT options are passed on through option files. If you specify "<modelname>.optfile = 1;" before the SOLVE statement in your GAMS model. CONVERT will then look for and read an option file with the name *convert.opt* (see "Using Solver Specific Options" for general use of solver option files). The syntax for the CONVERT option file is

```
optname value
```

with one option on each line. For example,

```
ampl
```

This option file would tell CONVERT to produce an AMPL input file. For file format options, the user can specify the filename for the file to be generated. For example, the option file entry

```
lingo myfile.lng
```

would generate a LINGO input file format called `myfile.lng`. Using the option `lingo` by itself, would produce the default output file for that option (`lingo.lng`).

All available options are listed in the following table.

Option	Description	Default
all	Generates all supported file formats.	
AlphaECP	Generates AlphaECP input file.	alpha.ecp
Ampl	Generates AMPL input file.	ampl.mod
AmplNLC	Generates Ampl NLC compatible file.	amplnlc.c
Baron	Generates BARON input file.	gams.bar
ConeReform	Reformulation of cone =C= constraints to NLP format. 0: keep conic =C= format 1: convert conic constraints to NLP format	0
CplexLP	Generates CPLEX LP format input file.	cplex.lp
CplexMPS	Generates CPLEX MPS format input file.	cplex.mps
Dict	Convert to GAMS dictionary.	dict.txt
FileList	Generates file list of file formats generated.	file.txt
FixedMPS	Generates fixed format MPS file.	fixed.mps
Gams	Generates GAMS scalar model. This is the default conversion format used.	gams.gms
Jacobian	Writes GDX version of current point.	jacobian.gdx
GmsInsert	Inserts the line `$if NOT '%gams.u1%' == '' $include'%gams.u1%'` before the solve statement.	
help	Generates option summary.	
include <filename>	Start reading from a new file.	
Lago	Generates a partial Lago file.	lago.gms
Lgo	Generates an LGO FORTRAN file.	lgomain.for
LindoMPI	Generates Lindo MPI file.	lindo.mpi
Lingo	Generates Lingo input file.	lingo.lng
LocalSolver	Generates a LocalSolver input file. This option is only available with ConvertD.	localsolver.lsp
match	Force a complete match for all MCP variable / equation pairs.	
memo	Generates a memo file containing model statistics and files created .	memo.txt
Minopt	Generates Minopt input file.	minopt.dat
NLP2MCP	Generates GAMS scalar MCP model.	gamsmcp.gms
ObjVar	Name of objective variable. By default the objective variable is just named via index, for example x1.	
Reform	Force reformulations.	100
Terminate	Force GAMS to terminate after conversion.	
ViennaDag	Generates Vienna Dag input file.	vienna.dag