Combining sets from multiple GDX files

28 November 2012

Introduction

It is very common to run a model many times, once for each of many scenarios, and write the results from each model run to a GDX file. The resulting GDX files can then be merged into a single GDX file that can be used as the input to a report writing process that generates reports on all scenarios in one step.

However, this seemingly simple process can be complicated if the set membership changes with each scenario. For example, consider running vSPD, say, for 30 days in a row. It is easy to imagine that in each trading period in each of the 30 days, the set of nodes or branches or offered units might be different. For report writing purposes, a 'super set' is desirable. That is, a set containing a unique listing of the entire union of set elements defined over all trading periods and days. Hence, the super set can be used as the domain for parameters read into GAMS from the merged GDX file, ensuring all data for all scenarios is loaded from the merged GDX file.

This note is intended to accompany a collection of small GAMS programs that demonstrate how to do this.

Worked GAMS code examples

The following GAMS programs should each be executed in the same order as they are described below to demonstrate how to prepare reports based on results from multiple runs of a GAMS-based model. Comments explaining what is being done at each step are included in each of the programs.

Example1.gms

This program creates two sets, *i* and *j*, and three parameters, *a*, *b* and *c*. The domain of *a* is set *i*, the domain of *b* is set *j*, and the domain of *c* is both sets *i* and *j*. All five symbols (sets and parameters) are written to a GDX file called *ex1_everything.gdx* and a subset of symbols is written to a GDX file called *ex1.gdx*.

In practical modelling applications, it is helpful to collect all output from a model run in a single GDX file for archive purposes, e.g. *ex1_everything.gdx*. At some future date, data can be extracted from this file without the need to re-run the model. At a minimum, the archive GDX should contain all sets and parameters used in the model, all variable levels and all equation marginal values. Note that many symbols in a GAMS program are used as an intermediate step to create the parameters used in the model – these symbols probably don't need to be archived.

One or more additional GDX files should be created to collect the information used to generate reports, e.g. *ex1.gdx* in the present case. The advantage of this approach is that the report writing process makes use of much smaller files. Similarly, if they need to be distributed by email or published on the web, it is convenient to be working only with those symbols actually needed. It is nearly always the case in any realistic modelling application, report writing requires only a tiny fraction of all the symbols created to formulate, parameterize and solve the model.

Example2.gms

This program is very similar to *example1.gms*; the key difference is that membership of the sets *i* and *j* is different. Actually, there is some overlap in the set membership across the two programs. As with *example1.gms*, e*xample2.gms* creates two GDX files – *ex2_everything.gdx* and *ex2.gdx*.

combineSets.gms

This program reads in sets *i* and *j* from both *ex1.gdx* and *ex2.gdx*. In the first instance, it reassigns the sets to new symbols, *ii* and *jj*. It combines the elements from each case to form a new 'super set' and then writes these combined super sets to a GDX file called *combinedSets.gdx*. In the process of writing the GDX file, it changes the set names back to *i* and *j*.

mergeGDXfiles.gms

This program creates and executes a batch file that copies *ex1.gdx* and *ex2* to *sc1.gdx* and *sc2.gdx*, respectively, and then merges *sc1.gdx* and *sc2.gdx* to create a yet another GDX file called *mergedResults.gdx*.

This step may appear trivial. However, it is a crucial step in being able to combine results from all scenarios into a single GDX file containing the same number of symbols as each individual scenario GDX file.

While in the case of just two scenarios this may seem insignificant, it is very powerful when there are many scenarios. Furthermore, every assignment statement in a GAMS report writing program can now be applied to all scenarios at once. In other words, there is no need to execute the report writing code once per scenario and then somehow (Excel vlookup?) join all the results together into a single table.

The key to this process is the creation of the scenario set, *sc*, in *mergeGDXfiles.gms*. The choice of element labels in this set is deliberate – the first element is *sc1* and the second is *sc2*. These labels are in fact the names given to the GDX files to be merged. The reason for this is that all symbols in the merged GDX file acquire a new domain index and the name of that domain is taken from the name of the files being merged.

For example, *ex1.gdx* (copied to *sc1.gdx*) and *ex2.gdx* (copied to *sc2.gdx*) each contained a symbol called *c*, which was defined on sets *i* and *j*. The merged file, *mergedResults.gdx*, also contains the symbol called *c*. But note how it is defined not only on sets *i* and *j*, but also on *sc*.

Example3.gms

This program reads in the relevant data from *combinedSets.gdx* and *mergedResults.gdx*. It can be used as the beginning of a report writing program.

A more sophisticated approach

All of the above can be made more sophisticated if the model is solved inside a loop on the scenario set. The GAMS *put_utility* can be used to write the GDX files each time around the loop and the GDX files would take their name from the text labels (*.tl*) of the elements assigned to set *sc*. The syntax would be something like this:

```
Set sc / sc1, sc2, etc, etc / ;
file dummy ;
loop(sc,
  ..
  .. code to assign parameter values for this solve of the model
  ..
  Solve vSPD minimising TOTALCOSTS using lp ;
  ..
  .. more code (maybe) to do post-solve computations on model output
  ..
  put dummy ;
  put_utility 'gdxout' / sc.tl ;
  execute_unload i j c or whatever it is you want in the GDX files to be merged
);
execute gdxmerge sc*.gdx output=mergedResults.gdx
```