Keep the Model Hot
A Scenario Solver for GAMS

Michael R. Bussieck
GAMS Development Corp.
MBussieck@gams.com
http://www.gams.com
GAMS at a Glance


Design Principles:

• Balanced mix of declarative and procedural elements
• Open architecture and interfaces to other systems
• Different layers with separation of:
  – model and data
  – model and solution methods
  – model and operating system
  – model and interface
Outline

- Prelude
- GAMS Execution System
- Scenario Solver
- GMO + Python & Co
Calling GAMS from your Application

Creating Input for GAMS Model
→ Data handling using GDX API

Callout to GAMS
→ GAMS option settings using Option API
→ Starting GAMS using GAMS API

Reading Solution from GAMS Model
→ Data handling using GDX API
Automated Generation of APIs

‘The GAMS Wrapper’

- API is defined using the GAMS language
- A tool written in GAMS is used to regenerate APIs for all languages
- Executed on request and nightly

→ A change in the definition of the API immediately makes it into all language interfaces
→ No manual and therefore error-prone efforts required
Automated Generation of APIs

‘The GAMS Wrapper’

- Automated nightly testing
- API version checks
  - Reusable for multiple GAMS component libraries
    - GAMS
    - GDX
    - Option
    - ...

Distributed GAMS APIs

- Component Libraries
  - GAMS
  - GDX
  - Option

- Supported languages
  - C, C++, C#
  - Delphi
  - Fortran
  - Java
  - VBA, VB.Net
  - Python

- Examples/Documentation
GAMS Execution System

- Mix of declarative and procedural language elements
  - Multiple models
  - Loops/if-then-else
  - Well suited for decomposition approaches (SP)
  - Many examples in the GAMS Model library
    - Cutting stock
    - TSP
    - ...
Loop(s,
    d(i,j) = dd(s,i,j);
    f = ff(s);
    solve mymodel min z using lp;
    rep(s) = mymodel.objval;
);

<table>
<thead>
<tr>
<th>Setting</th>
<th>Solve time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solvelink=0 (<em>default</em>)</td>
<td>40.297</td>
</tr>
<tr>
<td>Solvelink=%Solvelink.LoadLibrary%</td>
<td>03.625</td>
</tr>
</tbody>
</table>
GAMS Scenario Solver

cost.. z=e=sum((i,j),f*d(i,j)/1000*x(i,j));
set dict / s.scenario.'''
   d.param .dd
   f.param .ff
   x.level .xx /
solve mymodel min z using lp scenario dict;

• Update model data instead of matrix coefficients/rhs
• Hot start (keep the model hot inside the solver and use solver’s best update mechanism)
• Save model generation and solver setup time
• Model rim unchanged from scenario to scenario
• Apriori knowledge of all scenario data
• Dynamic model – rolling horizon

• Example:
  – Combined Heat and Power Planning with Heat Storage. All data known apriori but heat storage level.
  – Can’t use GAMS Scenario Solver
  – Implement Scenario Solver in Python
    • Identify some parameters as “modifiable” parameters
    • Implement rolling horizon in Python
GMO – GAMS Modeling Object

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Support shared-library implementation of solver links
- Support multiple models
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Implement once, run everywhere (multiple platforms & multiple languages)
- Comprehensive – one-stop shop for all linking needs

GMO Talk by Steve Dirkse, TC40, 13:30-15:00
Python & Co with GMO

- Populated GMO object (e.g. by GAMS)
- GMO API to allow modification and alteration of bounds, rhs, “modifiable” parameters (NL expression evaluation)
- GMO/GEV (GAMS Environment Object) based solver links
- Runtime system (Python, Java, …)

Alternative way to implement decomposition, and other algorithmic ideas based on MP models.

- Examples:
  - TSP (subtour elimination constraint generation)
  - Markowitz portfolio optimization
Summary

• Automated API generation for various GAMS components (GAMS, GDX, OPT) in various languages including Python.
• GAMS Scenario Solver approach for solving very similar models.
• GMO + Python & Co represent an alternative to GAMS execution system.

• Outlook: Get some of the improvements of the GMO + Python & Co approach back into GAMS.
• (Python) API for GMO not published yet (available on request). Still unclear where this experiment will lead us.
Contacting GAMS

Europe

GAMS Software GmbH
Eupener Str. 135-137
50933 Cologne
Germany

Phone: +49 221 949 9170
Fax: +49 221 949 9171
http://www.gams.de

info@gams.de

USA

GAMS Development Corp.
1217 Potomac Street, NW
Washington, DC 20007
USA

Phone: +1 202 342 0180
Fax: +1 202 342 0181
http://www.gams.com

sales@gams.com
support@gams.com