Enhanced Model Deployment and Solution in GAMS

Steve Dirkse
Introduction

• User interaction provided valuable feedback on:
  • The GAMS IDE
  • Building algorithms (decomposition, linearization) with GAMS code
  • Specific user: BEAM-ME project

• We took this feedback to heart: influenced recent development efforts
  • New user interface: GAMS Studio
  • Embedded Python
  • Updates to Object-Oriented API for Python
  • R/Shiny interface

• Enhanced model deployment and solution
GAMS Studio

- Open source Qt project (Mac/Linux/Win)
  - Published on GitHub under GPL
- First released in May 2018: focus on core functionality
  - Included most/best of the old IDE
  - Some additional features (e.g. column filters in GDX browser)
- Current projects now include new features (e.g. model inspector)
- Philosophy: release early to increase feedback for dev team
- Parallel release in near term: both Studio and IDE
Embedded Python: Why?

- GAMS is built for modeling
  - Syntax for parallel assignment and equation definition is compact, elegant, and efficient
  - Relational data model supports this – well-suited for the task
  - Traditional data structures not needed or available in GAMS – lists, trees, graphs, dictionaries
- String manipulation: useful for massaging data
- Plotting, map integration, other data visualization
- Sorting, permutations, randomization
- Specialized tasks: shortest path, factorization, subtour and cut generation, etc.
- Even more specialized: lexxing and parsing
Split Example – Data

country / system.empty /
city / system.empty /
mccCountry(cc,country)
mccCity (cc,city);
Split Example – Embedded Code

```python
onEmbeddedCode Python:
country = set()
city = set()
mccCountry = []
mccCity = []
for cc in gams.get("cc"):
    r = str.split(cc, " - ", 1)
country.add(r[0])
city.add(r[1])
mccCountry.append((cc,r[0]))
mccCity.append((cc,r[1]))
gams.set("country",list(country))
gams.set("city",list(city))
gams.set("mccCountry",mccCountry)
gams.set("mccCity",mccCity)
offEmbeddedCode country city mccCountry mccCity
```
GAMS ModellInstance

- demand(j) .. sum(i, x(i,j)) =g= bmult * b(j) ;

# assumes GamsCheckpoint cp
mi = cp.add_modelinstance()
bmult = mi.sync_db.add_parameter("bmult",0)
mi.instantiate("mus lp min z", GAMSModifier(bmult))
for b in myMultipliers:
    bmult.first_record().value = b
mi.solve()
Using R/Shiny to deploy GAMS models

- Data exchange via local files or database connection
- Modification and visualization of input data
- From a GAMS model to the first interface within minutes
- Comprehensive configurability
GENERAL ALGEBRAIC MODELING SYSTEM

Visualization

- Extensive visualization options
- Easily extendable with the wide spectrum of the R programming language.
Multiuser, multi-application support

- Local or server-based solution
- User authentication (e.g. LDAP, Keycloak, Google, Github, Facebook)
- Multi-Application support with docker-based technology
BEAM-ME Project

- Project goal: solve large Energy System Models (ESM): LPs
  - Block-diagonal structure with linking constraints and linking vars
  - Start with PIPS-IPM solver, extend as needed (e.g. linking constraints)
  - Use massively parallel hardware (thousands of 24-core nodes)
  - Do the modeling in GAMS
- Use distributed block-wise model generation:
  - Cut time and memory usage
  - Avoid limitation on model size (maxNNZ is ~2.1e9)

A PROJECT BY

[Logos of the institutions involved]
BEAM-ME Project: benefits

- Embedded Python – this was developed in large part for this project or with this project’s needs in mind
- Internal limits removed or relaxed
  - Internal data structures for string storage
  - Size limits imposed by 32-bit array offsets
  - Similar limits for tools and utilities included with GAMS
- Internal organization improved
  - Support for parallel model generation (in special cases)
  - GAMS/MPI – parallel GAMS runs synchronized with MPI
  - Execution-time save facility
Live Demo

- Requires recent GAMS version (25.0.2), current 25.1 even better
- Uses models from GAMS Model Library
- spbenders1 – *Benders example in GAMS, sequential solves, full regen*
- spbenders2 – *submodels solved in parallel in GAMS loop*
  - subproblem.solveLink = %solveLink.aSyncThreads%;
- spbenders3 - *sequential solves, Python modelinstance, no regen*
- spbenders4 – *parallel solves inside GAMS, full regen*
  - via mpi4py
- spbenders5 – *parallel solves, Python modelinstance, no regen*
Thank You