GAMSWorld and the Growing Demand for Reproducible Computational Experiments

Steve Dirkse
GAMS Development Corporation
www.gams.com

ICS Charleston    12 January  2009
Welcome/Agenda

Problems & Pitfalls
Solution: Reproducible Studies
GAMSWORLD components

Joint work with A Preussner, S Vigerske, M Bussieck
Vendor Performance Claims

- New & Improved!! 40% faster than the previous version!!
- New company claims: “faster and more robust”
- Gunslinger mentality: fastest Simplex in the West.
- Most of us fall prey to this mentality
Published Computational Results

- Algorithmic research: theory and computation
  - The theory can be verified
  - What about the computation?
- Some computational studies are very well done, others less so. How can we know which is which?
Reuse of models/data is difficult

- Once a study is done, the inputs are not preserved.
- Permission to share may be required
- Many formats make communication difficult.
- Provider has little incentive/priority to share
Implementors need real models

- Real models = actual applications from users
- Refinement is typically done using user model libs
- Improvements are targeted at weak spots in existing codes identified by tests on user models
- Robustness testing also done using real models – it is amazing what real users think to try!
Problems & Pitfalls

Solution: Reproducible Studies

GAMSWorld components
Solution: Reproducible studies

- Inputs: public test libraries available for download.
- Input translation: automated, using available tools
  - Many input formats, necessary in some cases
  - Worth avoiding if possible
- Result generation and collection
- Optional: independent validation
- Analysis & Visualization of the results
- Reproducibility implies automation: recurring motif
Vendor Performance Claims (2)

- New & Improved!! 40% faster than the previous version!!
- New company claims: “faster and more robust”
- Gunslinger mentality: fastest Simplex in the West.
- Most of us fall prey to this mentality

- Reproducible tests help by
  - Demonstrating what is faster (and what is not)
  - Allowing independent verification of claims
Published Computational Results (2)

- Algorithmic research: theory and computation
  - The theory can be verified
  - What about the computation?
- Some computational studies are very well done, others less so. How can we know which is which?

- Reproducible tests help by
  - Allowing independent verification of claims
  - Improving subsequent research, since the current status is better understood.
- New journal **MP Computation** will include evaluation and testing of software in the review process
Reuse of models/data is difficult (2)

• Once a study is done, the inputs are not preserved.
• Permission to share may be required
• Many formats make communication difficult.
• Provider has little incentive/priority to share

• Reproducible tests help by
  – Using test libraries that are and stay available
  – Using translation tools to share across different formats
  – Providing incentive to share models via test libraries
Implementors need real models (2)

- Real models = actual applications from users
- Refinement is typically done using user model libs
- Improvements are targeted at weak spots in existing codes identified by tests on user models
- Robustness testing also done using real models – it is amazing what real users think to try!

- Reproducible tests help by
  - Using test libraries that are and stay available
  - Analyzing/visualizing the results
Problems & Pitfalls
Solution: Reproducible Studies
GAMSWorld components
GAMSWorld Overview

- Bridges the academia-industry gap
- Collaborate on subjects of shared interest
- QA/Performance a large part of this effort
- Noteworthy items
  - Collaboration raises the profile, the stakes, the enjoyment
  - Automation is a recurring theme – no drudgery, no fudge factor
Welcome to the Performance World!

Performance World is a forum for discussion and dissemination of information and tools about all aspects of performance testing of solvers for mathematical programming problems. This world has been established in response to user demands for independent and reproducible performance results.

Overall performance highly depends on problem formulation, solver, and tuning parameters. Our performance tools are designed to serve the different needs of our user community. One user may be interested in finding the most reliable way to solve a proprietary or classified model. On the other hand, an academic researcher may be interested in testing a new algorithm against a set of existing test problems and competing approaches. The main features are:

- Uniform access to a comprehensive set of established and new test problems
- Automation tools for collecting performance measurements
- Tools for analyzing and visualizing test results

What's New:

- Several new libraries (Fixed Cost Network Flow and the Princeton NLP collection) have been added to the Performance Libraries
- A collection of quadratically constrained programs (QCP) have been added to the LINLIB set of models.
- The paper A Server for Automated Performance Analysis and Benchmarking of Optimization Software is available which includes an NLP benchmark using PAVER. See the results on all models or a subset of models.
Performance Libraries

- PerformanceLib contains multiple libraries
  - Different model types: LP, MIP, NLP, MINLP
  - Different sources: clients, literature, other libraries
- Available as problem instances, not applications
  - Helps disguise the source applications
  - Proper input when doing solver tests
Translation services

PAVER - GAMS Model Translation Web Submission Tool (GMS2XX)

The PAVER GAMS model translation web-submission tool runs the [GAMS/CONVERT] "solver" to translate GAMS models into the following languages:

- AlphaECP
- AMPL
- AmpInLC
- BARON
- CoinFML
- CplexLP
- CplexMPS
- Dict
- FixedMPS
- GAMS (scalar)
- Jacobian
- Lago
- Lgo
- LindoMPI
- LINGO
- MINOPT
- NLP2MCP
- ViennaDag
- ALL (this creates scalar versions of all supported languages, listed above)
Generating/Collecting Results

- Generate scripts to run tests
  - Includes commands to collect results via trace files
  - Model libraries include tools/data for this
- Run the tests – have coffee/lunch/sleep/vacation
- Optional: examine/verify the results

- This phase is user-dependent
  - A non-GAMS way could be quite different
- Maintain common inputs & outputs
GAMS/Examiner

- Purpose: to make an unbiased, independent report on the merit of points
- Points may come from GAMS or a solver
  - GAMS passes the previous solution as initial iterate
  - Solvers pass solutions back to GAMS
- Useful during solver debugging – helps pinpoint problems
  - Most checks are obvious – almost insultingly so
  - Updated as new ways to pass bad points are discovered
- Does checks on the scaled and unscaled (original) model
- All solution tolerances can be adjusted, default is tight
- Different points can be checked – no obvious right choice
Analysis/Visualization of results

- Done via PTOOLS
- Inputs text files (CSV), outputs HTML
- Tools can be run online or locally
- Robustness tools: Solver Square Utility, a cross-comparison of different solver outcomes
- Efficiency/speed tools
  - Time summary: cross-comparison of time used
  - Performance profiles
Open Testing Architecture

Can use Performance World tools

GAMS/Convert

Solve with “other” systems

GAMS Models

Web

Solve with GAMS

PAVER Server

Web

I. Models

II. Data Collection

III. Analysis & Visualization
Performance Tools Demo

- file:///C:/charleston/minlp/local_MINLP%20benchmarks%20from%2029.09.2008.htm
- Demo not working? See the next slides.
- Also see http://www.coin-or.org/GAMSlinks/benchmarks/
**PAVER: Solver Square**

**Result Totals in Percent:**

<table>
<thead>
<tr>
<th>Solver</th>
<th>% models optimal</th>
<th>% models feasible</th>
<th>% models infeasible</th>
<th>% models unbounded</th>
<th>% models fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver A</td>
<td>27.27</td>
<td>71.72</td>
<td>1.01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Solver B</td>
<td>-</td>
<td>57.60</td>
<td>0.00</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Result Totals in Number of Models:**

<table>
<thead>
<tr>
<th></th>
<th>optimal</th>
<th>feasible</th>
<th>infeasible</th>
<th>unbounded</th>
<th>fail</th>
<th>total Solver A</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>feasible</td>
<td>-</td>
<td>67</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>71</td>
</tr>
<tr>
<td>infeasible</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>unbounded</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>fail</td>
<td>-</td>
<td>19</td>
<td>6</td>
<td>-</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>total Solver A</td>
<td>-</td>
<td>87</td>
<td>8</td>
<td>-</td>
<td>4</td>
<td>99</td>
</tr>
</tbody>
</table>
Solver Resource Times

- Models for each solver pair outcome. Listed are the solver resource times in seconds, as well as the ratio of resource times for the two solvers if both solved optimally.
- Also listed are the objective values using both solvers. The better solution found is listed in boldface. A solution is considered better, if the relative objective function difference is greater than 1.00E-05.
- Solver resource time ratios for a particular model are listed only if one solver has resource greater than 5.00E-02.

<table>
<thead>
<tr>
<th>Modelname</th>
<th>Solver A</th>
<th>Solver B</th>
<th>Ratio (Solver A/Solver B)</th>
<th>Obj (Solver A)</th>
<th>Obj (Solver B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alan</td>
<td>0.0973</td>
<td>0.0100</td>
<td>9.730</td>
<td>3.600000000</td>
<td>2.925000000</td>
</tr>
<tr>
<td>batch</td>
<td>0.2478</td>
<td>0.5100</td>
<td>0.486</td>
<td>285506.50024405</td>
<td>285506.50000000</td>
</tr>
<tr>
<td>batchdes</td>
<td>0.1094</td>
<td>0.0400</td>
<td>2.735</td>
<td>167427.65714700</td>
<td>167427.70000000</td>
</tr>
<tr>
<td>du-opt</td>
<td>1.9718</td>
<td>0.5200</td>
<td>3.792</td>
<td>31.02527833</td>
<td>3.556340000</td>
</tr>
<tr>
<td>du-opt5</td>
<td>2.0975</td>
<td>1.7000</td>
<td>1.234</td>
<td>40.77273140</td>
<td>8.073658000</td>
</tr>
<tr>
<td>eg_all_s</td>
<td>28.3586</td>
<td>19.7100</td>
<td>1.437</td>
<td>11.23946600</td>
<td>7.920102000</td>
</tr>
<tr>
<td>eg_disc2_s</td>
<td>63.1667</td>
<td>5.3400</td>
<td>11.829</td>
<td>6.92006923</td>
<td>5.642101000</td>
</tr>
<tr>
<td>eg_disc_s</td>
<td>88.8061</td>
<td>9.3800</td>
<td>9.448</td>
<td>10.4217936</td>
<td>5.760540000</td>
</tr>
<tr>
<td>eg_int_s</td>
<td>106.3869</td>
<td>7.7900</td>
<td>13.657</td>
<td>7.88724302</td>
<td>7.463000000</td>
</tr>
<tr>
<td>elf</td>
<td>0.0573</td>
<td>15.3200</td>
<td>0.004</td>
<td>1.675000000</td>
<td>0.191666700</td>
</tr>
<tr>
<td>ex1221</td>
<td>0.0270</td>
<td>0.0000</td>
<td>---</td>
<td>7.667180000</td>
<td>7.667180000</td>
</tr>
<tr>
<td>ex1222</td>
<td>0.0629</td>
<td>999999.000000</td>
<td>0.000</td>
<td>1.07654308</td>
<td>1.07654300</td>
</tr>
</tbody>
</table>
### PAVER: Solver Resource Time

**Solvers used:**
- Solver A
- Solver B

**Modeltype(s):**
- MINLP

<table>
<thead>
<tr>
<th>Solver Condition</th>
<th>Total</th>
<th>Obj Solver A better</th>
<th>Obj same</th>
<th>Obj Solver B better</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver A infinitely faster</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Solver A much faster</td>
<td>13</td>
<td>1</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>Solver A faster</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Solvers perform the same</td>
<td>10</td>
<td>-</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>Solver B faster</td>
<td>31</td>
<td>-</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>Solver B much faster</td>
<td>12</td>
<td>-</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>Solver B infinitely faster</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>20</td>
</tr>
<tr>
<td>Both solvers failed to solve optimally</td>
<td>9</td>
<td>-</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>Total models:</td>
<td>99</td>
<td>5</td>
<td>47</td>
<td>47</td>
</tr>
</tbody>
</table>

This table compares solver performance across different model types and conditions, showing the number of models where each solver performs better, the same, or worse in terms of solve time. The table includes metrics for models solved infinitely faster, much faster, faster, and those that perform equally well or poorly.
### PAVER: Resource Time (cont.)

#### Solver Solver A much faster - Obj of Solver A better:

<table>
<thead>
<tr>
<th>Modelname</th>
<th>Solver A</th>
<th>Solver B</th>
<th>Ratio (Solver A / Solver B)</th>
<th>Obj Solver A</th>
<th>Obj Solver B</th>
</tr>
</thead>
<tbody>
<tr>
<td>synheat</td>
<td>0.2878</td>
<td>2.0600</td>
<td>0.140</td>
<td>1.549973335E+05</td>
<td>1.604355008E+05</td>
</tr>
</tbody>
</table>

#### Solver Solver A much faster - Obj same for both solvers:

<table>
<thead>
<tr>
<th>Modelname</th>
<th>Solver A</th>
<th>Solver B</th>
<th>Ratio (Solver A / Solver B)</th>
<th>Obj Solver A</th>
<th>Obj Solver B</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch</td>
<td>0.2478</td>
<td>0.5100</td>
<td>0.486</td>
<td>2.855065008E+05</td>
<td>2.855065008E+05</td>
</tr>
<tr>
<td>ex1222</td>
<td>0.0629</td>
<td>99999.000000</td>
<td>0.000</td>
<td>1.076543008E+00</td>
<td>1.076543008E+00</td>
</tr>
<tr>
<td>ex4</td>
<td>1.1326</td>
<td>3.8400</td>
<td>0.295</td>
<td>-8.06413616E+00</td>
<td>-8.064136008E+00</td>
</tr>
<tr>
<td>util</td>
<td>0.6693</td>
<td>14.2400</td>
<td>0.047</td>
<td>9.995787500E+02</td>
<td>9.995788000E+02</td>
</tr>
</tbody>
</table>
Profiles (best resource time)
Components/tools are downloadable:
- www.gamsworld.org/performance
- PAVER server: www.gamsworld.org/performance/paver

We ask & encourage you to use these tools
We are all better off sharing tools
Collaborators welcome!