Rapid Prototyping of Decomposition Algorithms

Part 1:
GAMS – Balancing Rapid Prototyping and High Performance

Michael Bussieck, GAMS Software GmbH, mbussieck@gams.com
Steffen Rebennack, Colorado School of Mines, srebenna@mines.edu
Agenda

- Introduction
- Declarative and Procedural Mix
- High Performance Prototypes
- Conclusion
Algebraic modeling language

From Wikipedia, the free encyclopedia

Algebraic Modeling Languages (AML) are high-level computer programming languages for describing and solving high complexity problems for large scale mathematical computation (i.e. large scale optimization type problems). One particular advantage of some algebraic modeling languages like AIMMS\textsuperscript{11}, AMPL\textsuperscript{2} or GAMS\textsuperscript{1} is the similarity of their syntax to the mathematical notation of optimization problems. This allows for a very concise and readable definition of problems in the domain of optimization, which is supported by certain language elements like sets, indices, algebraic expressions, powerful sparse index and data handling variables, constraints with arbitrary names. The algebraic formulation of a model does not contain any hints how to process it.

An AML does not solve those problems directly; instead, it calls appropriate external algorithms to obtain a solution. These algorithms are called solvers and can handle certain kind of mathematical problems like:

- linear problems
- integer problems
- (mixed integer) quadratic problems
- mixed complementarity problems
- mathematical programs with equilibrium constraints
- constrained nonlinear systems
- general nonlinear problems
- non-linear programs with discontinuous derivatives
- nonlinear integer problems
- global optimization problems
Algebraic Modeling Languages

What’s that?

http://en.wikipedia.org/wiki/Algebraic_modeling_language

- High-level computer programming languages for the formulation of complex mathematical optimization problems

- Notation similar to algebraic notation: Concise and readable definition of problems in the domain of optimization

- Do not solve problems directly, but ready-for-use links to state-of-the-art algorithms
General Algebraic Modeling System

The General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical optimization. GAMS is designed for modeling and solving linear, nonlinear, and mixed-integer optimization problems. The system is tailored for complex, large-scale modeling applications and allows the user to build large maintainable models that can be adapted to new situations. The system is available for use on various computer platforms. Models are portable from one platform to another.

GAMS was the first algebraic modeling language (AML) and is formally similar to commonly used fourth-generation programming languages. GAMS contains an integrated development environment (IDE) and is connected to a group of third-party optimization solvers. Among these solvers are BARON, COIN solvers, CONOPT, CPLEX, DICOPT, GUROBI, MOSEK, SNOPT, and XPRESS.

GAMS facilitates the users to implement a sort of hybrid algorithms combining different solvers in a seamless way. Models are described in concise algebraic statements which are easy to read, both for humans and machines. GAMS is among the most popular input formats for the NEOS Server for Optimization. Although initially designed for applications related to economics and management science, it has a large community of users from various backgrounds of engineering and science.

Contents

1 History
2 Timeline
3 Background

GAMS

Developer(s) GAMS Development Corporation
Stable release 23.7.3 / August 23, 2011
Development status Active
Platform Cross-platform
Type Algebraic Modeling Language (AML)
License Proprietary
Website GAMS USA
GAMS Germany

From Wikipedia, the free encyclopedia
GAMS Development / Software at a Glance

• Roots: World Bank, 1976
• Went commercial in 1987
• GAMS Development Corp. (US)
• GAMS Software GmbH (Europe)
• Technical tool provider (Software)
• Broad academic & commercial user community and network
  • GAMS is used in more than 120 countries
  • Half of licenses commercially used
Broad Network

ClustrMaps archive for http://www.gams.com/download/

5177 visits from 19 Mar 2012 to 26 Mar 2012

- distance in which individuals are clustered
- Total number of visits depicted above = 4275

Dot sizes:
- = 1000 +
- = 100 - 999
- = 10 - 99
- = 1 - 9
## Downloads (March 2012)

### Amazon CloudFront

Download Usage Report

<table>
<thead>
<tr>
<th>Region</th>
<th>Data Transfer Out</th>
<th>Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>United States</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.129 per GB - 1st TB / month data transfer out</td>
<td>197.126 GB</td>
</tr>
<tr>
<td></td>
<td>$0.0101 per 10,000 HTTPS Requests</td>
<td>3 Requests</td>
</tr>
<tr>
<td></td>
<td>$0.0075 per 10,000 HTTP Requests</td>
<td>52,154 Requests</td>
</tr>
<tr>
<td><strong>Europe</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.129 per GB - 1st TB / month data transfer out</td>
<td>212.982 GB</td>
</tr>
<tr>
<td></td>
<td>$0.0120 per 10,000 HTTPS Requests</td>
<td>1 Request</td>
</tr>
<tr>
<td></td>
<td>$0.00990 per 10,000 HTTP Requests</td>
<td>16,456 Requests</td>
</tr>
<tr>
<td><strong>Asia Pacific</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tokyo Region</td>
<td>$0.201 per GB - 1st TB / month data transfer out (includes consumption tax)</td>
<td>29.800 GB</td>
</tr>
<tr>
<td></td>
<td>$0.00995 per 10,000 HTTP Requests (includes consumption tax)</td>
<td>4,676 Requests</td>
</tr>
<tr>
<td>Singapore Region</td>
<td>$0.199 per GB - 1st TB / month data transfer out</td>
<td>39.512 GB</td>
</tr>
<tr>
<td></td>
<td>$0.012 per 10,000 HTTPS Requests</td>
<td>1 Request</td>
</tr>
<tr>
<td></td>
<td>$0.00990 per 10,000 HTTP Requests</td>
<td>18,087 Requests</td>
</tr>
<tr>
<td><strong>South America</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.259 per GB - 1st TB / month data transfer out</td>
<td>21,656 GB</td>
</tr>
<tr>
<td></td>
<td>$0.0160 per 10,000 HTTP Requests</td>
<td>1,535 Requests</td>
</tr>
</tbody>
</table>

**Total:** 495 GB ~ 5,500 monthly downloads
Agenda

- Introduction
- Declarative and Procedural Mix
- High Performance Prototypes
- Conclusion
What is a Model?

• List of Equations
  – *Mathematical Programming (MP) Model*

• Collection of several intertwined (MP) Models (polylithic models, Kallrath)
  – Data Preparation and Calibration
  – “Solution” Module
  – Reporting Module

• “Solution” Module often requires procedural/imperative programming
Declarative and Procedural Language Mix

• Declarative Elements in GAMS:
  – Model Algebra

• Procedural Elements in GAMS:
  – Programming Flow Control Features:
    • LOOP, FOR, WHILE, REPEAT
    • IF ELSEIF ELSE
    • Access to external programs/libraries
Cutting Stock Optimization at GSE

GSE-TRIM is a fully integrated module of the ERP-System GSE-PPS for Cutting Stock Optimization. Close cooperation of our in-house specialists with scientists in the area of discrete optimization has led to a number of successfully deployed applications used by the paper industry. Exact and hybrid optimization techniques coded in GAMS and Fortran have been implemented in our software package GSE-TRIM.

Our clients in various Mid-European paper industry companies benefit from:
- Exact waste minimization in roll production
- Non-standard objective functions
- Considering detailed operational restrictions
- Multi-stage format production

Based on a daily basis GSE-TRIM improves our clients key indicators and has been proven very stable over 7 years.

For more information please contact: www.gse-software.de

http://www.gams.com/presentations/index.htm#Ads
* Master model
Variable xp(p) patterns used
    z objective variable
Integer variable xp; xp.up(p) = sum(i, d(i));

Equation numpat number of patterns used
demand(i) meet demand;

numpat..  z =e= sum(pp, xp(pp));
demand(i).. sum(pp, aip(i,pp)*xp(pp)) =g= d(i);

model master /numpat, demand/;

* Pricing problem - Knapsack model
Variable  y(i) new pattern;
Integer variable y; y.up(i) = ceil(r/w(i));

Equation defobj
    knapsack knapsack constraint;

defobj..  z =e= 1 - sum(i, demand.m(i)*y(i));
knapsack.. sum(i, w(i)*y(i)) =l= r;

model pricing /defobj, knapsack/;
* Initialization - the initial patterns have a single width
pp(p) = ord(p)<=card(i);
aip(i,pp(p))$(ord(i)=ord(p)) = floor(r/w(i));
*display aip;

Scalar done  loop indicator /0/
Set     pi(p) set of the last pattern; pi(p) = ord(p)=card(pp)+1;

option optcr=0,limrow=0,limcol=0,solprint=off;

While(not done and card(pp)<card(p),
    solve master using rmip minimizing z;
    solve pricing using mip minimizing z;

* pattern that might improve the master model found?
    if(z.l < -0.001,
        aip(i,pi) = round(y.l(i));
        pp(pp) = yes; pi(p) = pi(p-1);
    else
        done = 1;
    );

display 'lower bound for number of rolls', master.objval;

option solprint=on;
solve masterprint using mip minimizing z;
Advantage of Algebraic Modeling System

Independence of
- Model and data
- Model and solution methods (solver)
- Model and operating system
- Model and user interface

→ Models benefit from
- Advancing hardware
- Enhanced / new solver technology
- Improved / upcoming interfaces to other systems
Introduction

Declarative and Procedural Mix

High Performance Prototypes

Conclusion
Simple Transport Model

Sets
  i  factories       /f1*f3/
  j  distribution centers /d1*d5/

Parameter
  capacity(i) /f1 500, f2 450, f3 650/
  demand(j)  /d1 160, d2 120, d3 270, d4 325, d5 700 /
  prodcost  unit production cost /14/
  price     sales price /24/
  wastecost cost of removal of overstocked products /4/

Table  transcost(i,j) unit transportation cost
   d1    d2    d3    d4    d5
  f1  2.49  5.21  3.76  4.85  2.07
  f2  1.46  2.54  1.83  1.86  4.76
  f3  3.26  3.08  2.60  3.76  4.45;
Simple Transport Model – Cont.

Variables
ship(i,j)  shipments
product(i)  units produced
received(j)  unit received
sales(j)  sales (actually sold)
waste(j)  overstocked products
profit

Positive variables ship, product, sales, waste;

Equations
obj
  production(i)
  receive(j)
  selling(j)
  market(j);

obj.. profit =e= sum(j, price*sales(j)) - sum((i,j), transcost(i,j)*ship(i,j))
  - sum(j, wastecost*waste(j)) - sum(i, prodcost*product(i));

production(i).. product(i) =e= sum(j, ship(i,j));
product.up(i) = capacity(i);

receive(j).. received(j) =e= sum(i, ship(i,j));
selling(j).. sales(j) =e= received(j) - waste(j);
market(j).. sales(j) =l= demand(j);

model transport /all/;
solve transport maximizing profit using lp;
Benders Decomposition for 2-Stage SP

Set
  s scenarios /lo,mid,hi/

* Stochastic demand plus probabilities
Table ScenarioData(s,*)
   d1  d2  d3  d4  d5  prob
lo  150 100 250 300 600  0.25
mid 160 120 270 325 700  0.50
hi  170 135 300 350 800  0.25;

\[
\begin{align*}
\min & \quad c^T x + \sum_{\omega} p(\omega) d^T_\omega y_\omega \\
\text{Ax} &= b \\
T_\omega x + W_\omega y_\omega &= h_\omega \\
x &\geq 0, y_\omega \geq 0
\end{align*}
\]

\[
\begin{align*}
\min & \quad d^T_\omega y_\omega \\
W_\omega y_\omega &= h_\omega - T_\omega \bar{x}^\nu \\
y_\omega &\geq 0
\end{align*}
\]

\[
\theta \geq \sum_{\omega \in \Omega} p_\omega \left( -\pi^{\ell}_\omega [T_\omega x + W_\omega \bar{y}_\omega^{\ell} - h_\omega] \right), \quad \ell = 1, \ldots, \nu - 1
\]

\[
x \geq 0
\]
Benders Decomposition for 2-Stage SP

```gams
loop(iter$(not done),
  * solve subproblems
    dyniter(iter) = yes;
    loop(s,
      demand(j) = ScenarioData(s,j);
      solve subproblem max zsub using lp;
      objsub(s) = zsub.l;
      cutconst(iter) = cutconst(iter) + p(s)*sum(j,market.m(j)*demand(j));
      cutcoeff(iter,j) = cutcoeff(iter,j) + p(s)*selling.m(j);
    );
    lowerbound = max(lowerbound, objmaster + sum(s, p(s)*objsub(s)));

  * convergence test
    if( (upperbound-lowerbound) < 0.001*(1+abs(upperbound)),
        done = 1;
    else
      * solve masterproblem
        solve masterproblem max zmaster using lp;
        upperbound = zmaster.l;
        objmaster = zmaster.l - theta.l;
    );
);```

Benders GAMS Implementation

- GAMS Implementation (solver Cplex)
  - 17 iterations: \((3+1)*17 = 68\) small models
  - 10.4 secs (all default)
  - 9.1 secs (minimize listing file size)
  - 4.9 secs (GAMS stays in memory)
  - 0.6 secs (communicate with solver through memory)

```plaintext
option limrow=0, limcol=0, solprint=silent, solvelink=%Solvelink.LoadLibrary%;
```

- Smart update of sub-model (Scenario Solver/GUSS)
- Grid computing
- Object Oriented API (e.g. .NET)
GUASS: Gather-Update-Solve-Scatter

cost..  \( Z = e = \text{sum}(i,j), f \times d(i,j) / 1000 \times X(i,j) \);

Loop(s,
  \( d(i,j) = dd(s,i,j) \);
  \( f = ff(s) \);
  \text{solve mymodel min } Z \text{ using } lp;
  \text{rep}(s) = Z.l;
);

set dict / s.scenario.''
  d.param .dd
  f.param .ff
  Z.level .rep /

solve mymodel min \( z \) using lp scenario dict;
* GUSS setup

Set dict / s. scenario. /
   demand. param. sDemand
   market. marginal. sMarket
   selling. marginal. sSelling
   zsub. level. objsub /

loop(iter$(not done),
  * solve subproblems
    dyniter(iter) = yes;
    solve subproblem max zsub using lp scenario dict;
    cutconst(iter) = cutconst(iter) + sum(s,p(s)*sum(j,sMarket(s,j)*sDemand(s,j)));
    cutcoeff(iter,j) = cutcoeff(iter,j) + sum(s,p(s)*sSelling(s,j));

    lowerbound = max(lowerbound, objmaster + sum(s, p(s)*objsub(s)));

  * convergence test
    if( (upperbound-lowerbound) < 0.001*(1+abs(upperbound)),
       done = 1;
    else
  * solve masterproblem
    solve masterproblem max zmaster using lp;
    upperbound = zmaster.l;
    objmaster = zmaster.l - theta.l;

); abort$(not done) "Too many iterations";
display zmaster.l, ship.l;
GUSS: Gather-Update-Solve-Scatter

<table>
<thead>
<tr>
<th>Setting</th>
<th>Solve time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solvelink=0 (default)</td>
<td>40.297</td>
</tr>
<tr>
<td>Solvelink=%Solvelink.LoadLibrary%</td>
<td>03.625</td>
</tr>
<tr>
<td>GUSS</td>
<td>00.797</td>
</tr>
</tbody>
</table>

- Updates model data instead of matrix coefficients/rhs
- Hot start (keeps the model hot inside the solver and uses solver’s best update mechanism)
- Saves model generation and solver setup time
- Transport model solution time: 0.3 secs
- A priori knowledge of all scenario data
demand=42; cost=14;
solve mymodel min obj using minlp;
report = var.l;
Parallel Power – GAMS Grid Facility

```gams
loop(scenario,
    demand=sdemand(scenario); cost=scost(scenario);
    solve mymodel min obj using minlp;
    report(scenario) = var.l);
```
Parallel Power – GAMS Grid Facility

mymodel.solvelink=3;
loop(scenario,
    demand=sdemand(scenario); cost=scost(scenario);
solve mymodel min obj using minlp;
h(scenario)=mymodel.handle);

Repeat
    loop(scenario$handlecollect(h(scenario)),
        report(scenario)=var.l;
        h(s) = 0);
        display$sleep(card(h)*0.02) 'sleep some time';
    until card(h)=0 or timeelapsed > 100;
dyniter(iter) = yes;

* Submission loop
loop(s,
    demand(j) = ScenarioData(s,j);
    solve subproblem max zsub using lp;
    h(s) = subproblem.handle;
);

* Collection loop
repeat
    loop(s$handlecollect(h(s)),
        objsub(s) = zsub.1;
        cutconst(iter) = cutconst(iter) + p(s)*sum(j,market.m(j)*sDemand(s,j));
        cutcoeff(iter,j) = cutcoeff(iter,j) + p(s)*selling.m(j);
        display$handledelete(h(s)) 'trouble deleting handles' ;
        h(s) = 0 );
    display$sleep(card(h)*0.02) 'was sleeping for some time';
    until card(h) = 0;
lowerbound = max(lowerbound, objmaster + sum(s, p(s)*objsub(s)));

* convergence test
if( (upperbound-lowerbound) < 0.001*(1+abs(upperbound)),


Object Oriented GAMS API

• High demand for OO API to GAMS
  – Embedding GAMS Model into IT infrastructure
  – GAMS .NET is currently in Alpha Client Testing
  – Java, Python, … will follow
  – OO API has the concept of a Model Instance
  – Build algorithms with GAMS objects in C#, Java, …

• GAMS solve statement
  – Update against the GAMS database (traditional)
  – Model Instance Object (new)
  – Use OO API to experiment
  – Introduction of Model Instance Object into GAMS
Agenda

- Introduction
- Declarative and Procedural Mix
- High Performance Prototypes
- Conclusion
Stochastic Programming with EMP

- EMP Information (stoch. Ext.)
- Original Model (det.)

Translation

Reformulated Model

Solving using Established algorithms

Solution

Mapping Solution into original space

Viewable
model transport /all/;
solve transport maximizing profit using lp;

file emp / '%emp.info%' /; put emp /* problem %gams.i%' /;
$onput
jrandvar demand('d1') demand('d2') demand('d3') demand('d4') demand('d5')
  0.25  150  100  250  300  600
  0.5  160  120  270  325  700
  0.25  170  135  300  350  800
$offput
putclose emp;

Set scen Scenarios / s1*s3 /;
Parameter
  sc_demand(scen,j) demand by scenario
  sc_sales(scen,j)  sales by scenario
  sc_waste(scen,j)  waste by scenario
  sc_profit(scen)   profit by scenario;

Set dict / scen .scenario.''
  demand .randvar. sc_demand
  sales .level. sc_sales
  waste .level. sc_waste
  profit .level. sc_profit /;

option emp=de;
solve transport maximizing profit using emp scenario dict;
display ship.l, sc_demand, sc_sales, sc_waste, sc_profit;
Yet Another Math Programming Consultant

So I am now a full time math programming consultant... I will try to post my (technical) notes here. Keeping a searchable list of them will make this useful for me in my daily life.

Sunday, April 15, 2012

Parallel GAMS Jobs (2)

In http://yetanothermathprogrammingconsultant.blogspot.com/2012/04/parallel-gams-jobs.html I described a simple approach I suggested to a client allowing to run multiple scenarios in parallel.

For a different client we needed to run a randomized algorithm that solves many small MIP models. They are so small that using multiple threads inside the MIP solver does not give much performance boost (much of the time is spent outside the pure Branch & Bound part - such as preprocessing etc.). However as the MIP problems are independent of each other we could generate all the necessary data in advance and then call the scenario solver (http://www.gams.com/modlib/adddocs/gusspaper.pdf). This will keep the generated problem in memory, and does in-core updates, so we don’t regenerate the model all the time.

The implementation does not win the beauty contest, but it could be developed quickly.

<table>
<thead>
<tr>
<th>Implementation</th>
<th># of MIP models</th>
<th>Solve time</th>
<th>Rest of algorithm</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional GAMS loop (call solver as DLL)</td>
<td>100,000</td>
<td>1068 sec</td>
<td>169 sec</td>
<td>1237 sec</td>
</tr>
<tr>
<td>Scenario Solver</td>
<td>100,000</td>
<td>293 sec</td>
<td>166 sec</td>
<td>459 sec</td>
</tr>
</tbody>
</table>

To get more performance I tried to run the scenario solver in parallel. That is not completely trivial as the solver has a number glitches (e.g. scratch files with fixed, hard coded names). I also run parts of the GAMS algorithm in parallel, but some parts had to be done in the master model after merging the results.

<table>
<thead>
<tr>
<th>Implementation</th>
<th># of MIP models</th>
<th>Worker threads</th>
<th>Parallel sub-problem time</th>
<th>Rest of algorithm (Serial)</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel + Scenario Solver</td>
<td>100,000</td>
<td>4</td>
<td>116 sec</td>
<td>67 sec</td>
<td>183 sec</td>
</tr>
</tbody>
</table>

The implementation does not win the beauty contest, but it could be developed quickly. For these larger...
Thank You!

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

Europe
GAMS Software GmbH
Eupener Str. 135-137
50933 Cologne
Germany
Phone: +49 221 949 9170
Fax: +49 221 949 9171

http://www.gams.com
info@gams.de
support@gams.com