GAMS – An Introduction

Hands-on Tutorial on Optimization

Frederik Proske & Lutz Westermann
GAMS Software GmbH
Agenda

GAMS at a Glance

GAMS – Hands On Examples

Outlook
  • A Model is a Model is a Model
  • Parallelization on HPC Systems
  • Deployment of GAMS Models
  • ...

GAMS at a Glance
1976 - A World Bank Slides

The Vision

GAMS came into being!

RESULT:
- Limited drain of resources
- Same representation of models for humans and machines
- Model representation is also model documentation
The aim of this system is to provide one representation of a model which is easily understood by both humans and machines.


Self-documenting model. A GAMS model is a machine-executable documentation of an optimization problem.

What did this give us?

- Simplified model development & maintenance
- Increased productivity tremendously
- Made mathematical optimization available to a broader audience (domain experts)

2012 INFORMS Impact Prize
Company

- Roots: World Bank, 1976
- Went commercial in 1987

Locations
- GAMS Development Corporation (USA)
- GAMS Software GmbH (Germany)

Product
- The General Algebraic Modeling System
Broad User Community and Network

- 14,000+ licenses
- Users: 50% academic, 50% commercial
- GAMS used in more than 120 countries
- Uniform interface to ~40 solvers

30+ Years GAMS Development
## Broad Range of Application Areas

<table>
<thead>
<tr>
<th>Agricultural Economics</th>
<th>Applied General Equilibrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical Engineering</td>
<td>Economic Development</td>
</tr>
<tr>
<td>Econometrics</td>
<td>Energy</td>
</tr>
<tr>
<td>Environmental Economics</td>
<td>Engineering</td>
</tr>
<tr>
<td>Finance</td>
<td>Forestry</td>
</tr>
<tr>
<td>International Trade</td>
<td>Logistics</td>
</tr>
<tr>
<td>Macro Economics</td>
<td>Military</td>
</tr>
<tr>
<td>Management Science/OR</td>
<td>Mathematics</td>
</tr>
<tr>
<td>Micro Economics</td>
<td>Physics</td>
</tr>
</tbody>
</table>
Foundation of GAMS

- Powerful algebraic modeling language
- Open architecture and interfaces to other systems, independent layers
Declarative and Procedural Language Elements

**Declarative elements**
- Similar to mathematical notation
- Easy to learn - few basic language elements: sets, parameters, variables, equations, models
- Model is executable (algebraic) description of the problem

**Procedural elements**
- Control Flow Statements (e.g. loops, for, if,...)
- Build complex problem algorithms within GAMS
- Simplified interaction with other systems
  - Data exchange
  - APIs
Independence of Model and Operating System

Platforms supported by GAMS:

- Models can be moved between platforms with ease!
Cross Platform GUI – GAMS Studio

- Open source Qt project (Mac/Linux/Win)
  - Published on GitHub under GPL
- Released in beta status
- Group Explorer
- Editor / Syntax coloring / Spell checks

- Tree view / Syntax-error navigation
- Option Editor
- Integrated Help
- Model Debugging & Profiling
- Solver selection & setup
- Data viewer
- GAMS Processes Control
Independence of Model and Solver

One environment for a wide range of model types and solvers

All major commercial LP/MIP solver

Open Source Solver (COIN)

Also solver for NLP, MINLP, global, and stochastic optimization

Switching between solvers with one line of code!
Independence of Model and Data

- Declarative Modeling
- ASCII: Initial model development

- GDX: Data layer ("contract") between GAMS and applications
  - Platform independent
  - No license required
  - Direct GDX interfaces and general API
  - ...
Independence of **Model and User Interface**

**API’s**

- *Low Level*
- **Object Oriented**: .Net, Java, Python, C++
- No modeling capability: Model is written in GAMS
- Wrapper class that encapsulates a GAMS model
Free Model Libraries

➢ More than 1400 models!
Why GAMS?

• Experience of 30+ years
• Broad user community from different areas
• Lots of model templates
• Strong development interface

• Consistent implementation of design principles
  • Simple, but powerful modeling language
  • Independent layers
  • Open architecture: Designed to interact with other applications

• Open for new developments
• Protecting investments of users
GAMS – Building blocks
Sets

- **Basic elements** of a model
- **Syntax:**
  ```
  set[s]                  
  set_name ["text"] [/element [text] {,element [text]} /]   
  {,set_name ["text"] [/element [text] {,element [text]} /]} ;
  ```
- **Elements** have up to 63 characters, start letter or digit or are quoted
- Elements have **no value** (!), that is, ‘1986’ does not have the numerical value 1986 and ‘01’ ≠ ‘1’
- Text has up to 254 characters, all in one line
- **Example:**

```plaintext
1  Set n  Nutrients
2 / Prot  "Protein (mg)"
3    VitA  "Vitamine A",
4       Calc Calcium
5    /;
```
Data

• Data in GAMS consists always of real numbers (no strings)
• Uninitialized data has the default value 0
• 3 forms to declare data:
  • Scalar: a single (scalar) number
  • Parameter: list oriented data
  • Table: table oriented data (at least 2 dimensions)

• Scalar Data:
  • Syntax:

```
scalar[s] scalar_name [text] [/numerical_value/]
  { scalar_name [text] [/numerical_value/]} ;
```

• Example:

```
1  Scalars  rho Discountfactor / .15 /
2        izf internal rate of return;
```
Data: Parameters

• Can be indexed over one or several sets
• Syntax:

```plaintext
parameter[s] param_name[(index_list)] [text] [{,element [=] numerical_value
{},element [=] numerical_value} /]
    {},param_name[(index_list)] [text] [{,element [=] numerical_value
    {},element [=] numerical_value} /] ;
```

• Example:

```plaintext
1 set ice icecreams / chocolate, strawberry, cherry, vanilla /;
2 parameter demand(ice) / chocolate 50, strawberry = 30
3          vanilla 20 /
```

• Example:

```plaintext
1 set c 'countries' / jamaica, haiti, guyana, brazil /;
2 parameter demand(c,ice) "Demand of icecream per country (t)"
4     Haiti.(Chocolate,Vanilla,Strawberry) = 30,
5     (Guyana,Brazil).Chocolate 100 /
```
Data: Table

- Syntax:

```
table table_name[(index_list)] [text] EOL
  element { element } EOL
  element numerical_value { numerical_value } EOL
{element numerical_value} EOL}
```

- Example:

```
1 table demand(c,ice) "Demand of icecream per country (t)"
2                               Chocolate Strawberry Cherry Vanilla
3 Jamaica                          300         50         5
4 Haiti                             30          30          30
5 (Guyana, Brazil)                 100          
```

- No “free form”: position of elements is of importance
- tables with more than 2 dimensions are also possible
Data: Assignments

• Scalar Assignment:

```plaintext
1 scalar x / 1.5 /;
2 x = 1.2;
3 x = x + 2;
```

• Indexed Assignment:

```plaintext
1 Set row / r1*r10 /;
2 col / c1*c10 /;
3 subrow(row) / r7*r10 /;
4 Parameter a(row,col), r(row), c(col);
5 a(row,col) = 13.2 + r(row)*c(col);
6 a('r7','c4') = -2.36;
7 a(subrow,'c10') = 2.44 - 33*r(subrow);
8 a(row,row) = 7.7 - r(row);
9 alias(row,rowp);
10 a(row,rowp) = 7.7 - r(row) + r(rowp);
```
Data: Expressions (I)

Expression: an arbitrarily complex calculation instruction

**Arithmetic operators:** +, -, *, /, ** (exponentiate)

1. \( x = 5 + 4*3**2; \)
2. \( x = 5 + 4*(3**2); \)

- \( x^n \) corresponds to \( \exp(n \ln(x)) \) -> only allowed for \( x > 0 \)
- \( \text{power}(x,n) \) can be used instead (if \( n \in \mathbb{Z} \))

\[
\text{population}(t) = 56*(1.015**(\text{ord}(t)-1))
\]
Indexed Operations:

- Syntax: indexed_op( (controlling_indices), expressions)
- indexed_op can be: `sum`, `prod`, `smin`, `smax`

```plaintext
1  parameter demand(c,ice)  "demand (t)"
2  totaldemand(c)  "totaler demand per country (t)"
3  completedemand  "totaler demand for all countries (t)"
4      mindemand(ice)  "minimal demand per icecream"
5  totaldemand(c) = sum(ice, demand(c,ice));
6  completedemand = sum((c,ice), demand(c,ice));
7      mindemand(ice) = smin(c, demand(c,ice));
```

Functions: erf(x), exp(x), power(x,n), sqr(x), uniform(a,b), normal(mean,sdev), ...
Variables

Syntax: `[var-type] variable[s] varname [text] {, varname [text]}`

- var-type allows to pre-determine the Range of a variable:

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>free (default)</td>
<td>$\mathbb{R}$</td>
</tr>
<tr>
<td>positive</td>
<td>$\mathbb{R}_{\geq 0}$</td>
</tr>
<tr>
<td>negative</td>
<td>$\mathbb{R}_{\leq 0}$</td>
</tr>
<tr>
<td>binary</td>
<td>{0, 1}</td>
</tr>
<tr>
<td>integer</td>
<td>{0, 1, ..., 100} (!!)</td>
</tr>
<tr>
<td>semiconst</td>
<td>{0} $\cup {\ell, u]$ (default: $\ell = 1, u = \infty$)</td>
</tr>
<tr>
<td>semiint</td>
<td>{0} $\cup {\ell, \ell + 1, \ldots, u}$ (default: $\ell = 1, u = 100$)</td>
</tr>
<tr>
<td>sos1, sos2</td>
<td>special ordered sets of type 1 and 2</td>
</tr>
</tbody>
</table>

Examples:

1. `variables x, y, objvar;`
2. `positive variable x;`
3. `integer variable z;`
Variable Attributes

Attributes of a variable:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.lo</td>
<td>lower bound on variable range</td>
</tr>
<tr>
<td>.up</td>
<td>upper bound on variable range</td>
</tr>
<tr>
<td>.fx</td>
<td>fixed value for a variable</td>
</tr>
<tr>
<td>.l</td>
<td>current primal value (updated by solver)</td>
</tr>
<tr>
<td>.m</td>
<td>current dual value (updated by solver)</td>
</tr>
<tr>
<td>.scale</td>
<td>scaling factor</td>
</tr>
<tr>
<td>.prior</td>
<td>branching priority</td>
</tr>
</tbody>
</table>

Examples:

1  \texttt{x.up} = 10;
2  \texttt{y.fx} = 5.5;
3  \texttt{display z.l;}
Equations (I)

- Equations serve to define restrictions (constraints) and an objective function.

**Declaration:**
- Syntax: `Equation[s] eqnname [text] {, eqnname [text]} ;`
- Example:

```plaintext
1 Equation objective "Objective Function";
```

**Definition:**
- Syntax: `eqnname(domainlist).. expression eqn_type expression;

<table>
<thead>
<tr>
<th>eqn_type</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=e=</td>
<td>=</td>
</tr>
<tr>
<td>=g=</td>
<td>≥</td>
</tr>
<tr>
<td>=l=</td>
<td>≤</td>
</tr>
<tr>
<td>[...]</td>
<td>[...]</td>
</tr>
</tbody>
</table>
Equations (II)

Example:

```plaintext
1  objective.. objvar =e= 2 * x + 3 * y * y - y + 5 * z ;
2  e1..       x + y   =e=  z;
```

Attributes of an equation:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.l</td>
<td>activity (updated by solver)</td>
</tr>
<tr>
<td>.m</td>
<td>current dual value (updated by solver)</td>
</tr>
<tr>
<td>.scale</td>
<td>scaling factor</td>
</tr>
</tbody>
</table>
Model Statements (I)

- Model = collection of equations
- Syntax:

```
model[s] model_name [text] [/ all | eqn_name {}; eqn_name /] }
{,model_name [text] [/ all | eqn_name {}; eqn_name /] }
```

- Example:

```
1  model m / all /;
2  model m / objective, e1 /;
```
Model Statements (II)

• Attributes:

<table>
<thead>
<tr>
<th>set by user</th>
<th>set by solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterlim  iteration limit</td>
<td>iterusd number of iterations</td>
</tr>
<tr>
<td>reslim time limit in seconds</td>
<td>resusd solving time</td>
</tr>
<tr>
<td>optcr relative gap tolerance</td>
<td>modelstat model status</td>
</tr>
<tr>
<td>optfile number of solver options file</td>
<td>solvestat solver status</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• Example:

```plaintext
m.reslim = 60; m.optcr = 0;
solve m minimize objvar using MINLP;
display m.resusd;
```
Solving a Model

- Passing a model to a solver and evaluation of results
- Specification of one free variable to be minimized or maximized
- Syntax:

```plaintext
1 solve modelname using modeltyp maximizing|minimizing varname;
2 solve modelname maximizing|minimizing varname using modeltyp ;
```

- the **model type** defines the problem class to be used for the model:
  - LP      a linear problem
  - QCP     quadratically constraint problem (only linear and quadratic terms)
  - NLP     a nonlinear problem with continuous functions
  - MIP     a mixed-integer linear problem
  - [...]

- Example:

```plaintext
1 solve m using MINLP minimizing objvar;
2 solve m using RMINLP min         objvar;
```
A Simple *Transportation Problem*

- What does this example show?
  - It gives a first glimpse of how a problem can be formulated in GAMS
  - It shows some basics of data exchange with GAMS
  - It shows how easy it is to change model type and, consequently, solver technology
LP
• Determine minimum transportation cost
• Result: city to city shipment volumes

MIP
• Discrete decisions
• E.g.: Ship at least 100 cases

MINLP
• Non-linearity
• E.g.: Decrease in unit cost with growing volumes

Scenarios
• SolveLink
• Grid Facility
• GUSS
A Simple Transportation Problem

Canning Plants (supply)  shipments  Markets (demand)

(Number of cases)

Freight: $90 case / thousand miles
A Simple Transportation Problem

Minimize Transportation cost
subject to
Demand satisfaction at markets
Supply constraints

Freight: $90 case / thousand miles

Supply (cases)
- Seattle: 350
- San Diego: 600

Distance (thousand miles)
- Seattle to Topeka: 1.8
- Seattle to Chicago: 1.7
- Seattle to New York: 2.5
- San Diego to Topeka: 1.4
- San Diego to Chicago: 1.8
- San Diego to New York: 2.5

Demand (cases)
- Topeka: 275
- Chicago: 300
- New York: 325

Supply constraints

Demand satisfaction at markets

Transportation cost
Mathematical Model Formulation

Indices:
- $i$ (Canning plants)
- $j$ (Markets)

Decision variables:
- $x_{ij}$ (Number of cases to ship)

Data:
- $c_{ij}$ (Transport cost per case)
- $a_i$ (Capacity in cases)
- $b_i$ (Demand in cases)

$$\min \sum_i \sum_j c_{ij} \cdot x_{ij} \quad \text{(Minimize total transportation cost)}$$

subject to
- $\sum_j x_{ij} \leq a_i \quad \forall i \quad \text{(Shipments from each plant} \leq \text{ supply capacity)}$
- $\sum_i x_{ij} \geq b_j \quad \forall j \quad \text{(Shipments to each market} \geq \text{ demand)}$
- $x_{ij} \geq 0 \quad \forall i, j \quad \text{(Do not ship from market to plant)}$
- $i, j \in \mathbb{N}$
GAMS Syntax (LP Model)

```
Variables
x(i,j) shipment quantities in cases
z total transportation costs in thousands of dollars;

Positive Variable x;

Equations
cost define objective function
supply(i) observe supply limit at plant i
demand(j) satisfy demand at market j;

cost .. z =e= sum((i,j), c(i,j)*x(i,j));
supply(i) .. sum(j, x(i,j)) =l= a(i);
demand(j) .. sum(i, x(i,j)) =g= b(j);

Model transportLP /all/;
Solve transportLP using lp minimizing z;
```
GAMS Syntax (Data Input)

```plaintext
Sets
  i  canning plants
  j  markets ;

Parameters
  a(i)  capacity of plant i in cases
  b(j)  demand at market j in cases
  d(i,j) distance in thousands of miles
  c(i,j) transport cost in thousands of dollars per case ;

$onecho > instructions.txt
par=d rng=Sheet1!A1 rdim=1 cdim=1
par=b rng=Sheet1!B6 rdim=0 cdim=1
par=a rng=Sheet1!G2 rdim=1 cdim=0
$offecho
$call gdxrw data.xlsx o=data.gdx @instructions.txt
$if errorlevel 1 $abort Error preparing data

gdxin data.gdx
$load i<d.dmi; j<d.dmi2; d a b

gdxin

Scalar f  freight in dollars per case per thousand miles /90/ ;
c(i,j) = f * d(i,j) / 1000 ;
```
Solution to LP model

Canning Plants (supply) ➔ shipments (Number of cases) ➔ Markets (demand)

Freight: $90 case / thousand miles

Total cost: $153,675
LP
• Determine minimum transportation cost
• Result: city to city shipment volumes

MIP
• Discrete decisions
• E.g.: Ship at least 100 cases

MINLP
• Non-linearity
• E.g.: Decrease in unit cost with growing volumes

Scenarios
• SolveLink
• Grid Facility
• GUSS
MIP Model: Minimum Shipment of 100 cases

- Shipment volume: \( x \) \hspace{1cm} \text{(continuous variable)}
- Discrete decision: \( \text{ship} \) \hspace{1cm} \text{(binary variable)}

Add constraints:

\[ x_{i,j} \geq 100 \cdot \text{ship}_{i,j} \quad \forall \ i,j \quad \text{(if ship=1, then ship at least 100)} \]

\[ x_{i,j} \leq \text{big}M \cdot \text{ship}_{i,j} \quad \forall \ i,j \quad \text{(if ship=0, then do not ship at all)} \]

\( \text{ship}_{i,j} \in \{0,1\} \)
MIP Model: **GAMS Syntax**

```gams
* MIP
scalar minS minimum shipment / 100 /
bigM big M;
bigM = min(max(a(i), b(j)));

binary variable ship(i,j) '1 if we ship from i to j, otherwise 0';

equation minship(i,j) minimum shipment
maxship(i,j) maximum shipment;

minship(i,j).. x(i,j) =g= minS * ship(i,j);
maxship(i,j).. x(i,j) =l= bigM * ship(i,j);

Model transportMIP / transportLP, minship, maxship /;
option optcr = 0;

Solve transportMIP using MIP minimizing z;
rep(i,j,'MIP') = x.l(i,j);
display rep;
```
MIP Model: Results
MIP Model: Solution

Canning Plants (supply) \[\rightarrow\] shipments (Number of cases) \[\rightarrow\] Markets (demand)

- Seattle (capacity: 350)
- Topeka (demand: 275)
- Chicago (demand: 300)
- New-York (demand: 325)
- San-Diego (capacity: 600)

Freight: $90 case / thousand miles

Total cost: $153,675
LP
• Determine minimum transportation cost
• Result: city to city shipment volumes

MIP
• Discrete decisions
• E.g.: Ship at least 100 cases

MINLP
• Non-linearity
• E.g.: Decrease in unit cost with growing volumes

Scenarios
• SolveLink
• Grid Facility
• GUSS
MINLP: Cost Savings

The cost per case decreases with an increasing shipment volume.

x (Number of cases)

Replace:

\[
\min \sum_i \sum_j c_{ij} \cdot x_{ij}
\]
(Minimize total transportation cost)

With

\[
\min \sum_i \sum_j c_{ij} \cdot x_{ij}^{\beta}
\]
(Minimize total transportation cost)
MINLP Model: GAMS Syntax

```gams
* MINLP
Scalar    beta / 0.95 /
Equation  costnlp define non-linear objective function;
costnlp.. z  =e=  sum((i,j), c(i,j)*x(i,j)**beta) ;
Model    transportMINLP / transportMIP - cost + costnlp /;
Solve    transportMINLP using MINLP minimizing z ;
rep(i,j,'MINLP') = x.l(i,j);
display  rep;
```
MINLP Model: Results
MINLP Model: Solution

Canning Plants (supply)  shipments (Number of cases)  Markets (demand)

Seattle (capacity: 350)  300  Chicago (demand: 300)
Topeka (demand: 275)  275  New-York (demand: 325)
San-Diego (capacity: 600)  325

Freight: $90 case / thousand miles  Total cost: $115,438
LP
- Determine minimum transportation cost
- Result: city to city shipment volumes

MIP
- Discrete decisions
- E.g.: Ship at least 100 cases

MINLP
- Non-linearity
- E.g.: Decrease in unit cost with growing volumes

Scenarios
- SolveLink
- Grid Facility
- GUSS
Motivation

• Solving challenging real-world problems often involves the solution of many optimization problems
  • Decomposition Methods
  • Scenario Analysis
  • Heuristics
  • …

• Such approaches are often chosen, if solving the problem at hand does not work with a single monolithic model, e.g.
  • Due to it’s size and the required resources (e.g. memory)
  • Due to time restrictions (Problem should be solved in minutes but it takes days)
  • …

→ GAMS Grid Facility
→ Gather-Update-Solve-Scatter
SolveLink Option

Controls GAMS function when linking to solver.

Model transport /all/ ;
transport.solveLink={0  %Solvelink.ChainScript%,
1  %Solvelink.CallScript%,
2  %Solvelink.CallModule%,
3  %Solvelink.AsyncGrid%,
4  %Solvelink.AsyncSimulate%,
5  %Solvelink.LoadLibrary%,
6  %solveLink.aSyncThreads%,
7  %solveLink.threadsSimulate%};
solve transport using lp minimizing z;
SolveLink Option – **Sequential Solves**

- **ChainScript [0]:** Solver process, GAMS vacates memory
  - Maximum memory available to solver
  - Protection against solver failure (*hostile* link)
  - Swap to disk

- **Call{Script [1]/Module [2]}:** Solver process, GAMS stays live
  - Protection against solver failure (*hostile* link)
  - No swap of GAMS database
  - File based model communication

- **LoadLibrary [5]:** Solver DLL in GAMS process
  - Fast memory based model communication
  - Update of model object inside the solver (hot start)
  - Not supported by all solvers
SolveLink Option Sequential – Exercise

• Generate 100 distance scenarios with random data
• Solve these scenarios with the solveLink values…
  • ChainScript [0]
  • CallModule [2]
  • LoadLibrary [5]
• Compare the execution time of solving all scenarios with different solveLink settings
  • Hint: Look at the GAMS function jNow
SolveLink Option – Sequential Solves

64  Model transport /all/ ;
65
66  set s  scenarios / s1*s100 /;
67  s1 solveLink / ChainScript, CallModule, LoadLibrary / ;
68  parameter dd(s,i,j)  distance by scenario
69      time(*)  time for 100 scenarios
70      sl_val(s) solveLink value / ChainScript %solveLink.chainScript%,
71         CallModule %solveLink.CallModule%,
72         LoadLibrary %solveLink.loadLibrary% / ;
73
74  scalar      tmp;
75
76  dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
77
78  option limrow=0, limcol=0, solprint=silent;
79
80  * SERIAL SOLVE
81  loop(s1,
82     tmp = jnow;
83     transport.solveLink=sl_val(s1);
84  loop(s,
85      d(i,j) = dd(s,i,j);
86      Solve transport using lp minimizing z ;
87  ) ;
88  time(s1) = (jnow-tmp)*24*60*60;
89 ) ;
90  display time;
91

----  88 PARAMETER time  time for 100 scenarios
ChainScript 6.710,  CallModule  2.694,  LoadLibrary  0.578
SolveLink Option – Asynchronous Solves

• aSyncGrid [3]: GAMS starts the solution and continues in a Grid computing environment

• aSyncThreads [6]: The problem is passed to the solver in core without use of temporary files, GAMS does not wait for the solver to come back
SolveLink Option Asynchronous – Exercise

• Generate 100 distance scenarios with random data
• Solve these scenarios with the solveLink values...
  • aSyncGrid [3]
  • aSyncThreads [6]
• Compare the execution time of solving all scenarios with different solveLink settings
  • Hint: Check the log for output about solveLink
  • → Use solver CplexD instead of Cplex
  • Hint: Look at the following GAMS functions:
    • readyCollect
    • handleCollect
    • handleDelete
SolveLink Option – Asynchronous Solves

Welcome

64 Model transport /all/;
65
66 set s scenarios / s1*s100 /
67 sl solvelink / aSyncGrid, aSyncThreads /;
68 parameter dd(s,i,j) distance by scenario
69   time(*) time for 100 scenarios
70   sl_val(sl) solvelink value / aSyncGrid %solveLink.aSyncGrid%,
71   aSyncThreads %solveLink.aSyncThreads% /
72
73 scalar tmp;
74
75 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
76 option limrow=0, limcol=0, solprint=silent, lp=cplexd;
77 *
78 loop(sl,
79   tmp = jnow;
80   transport.solvelink=sl_val(sl);
81
82   loop(s,
83     d(i,j) = dd(s,i,j);
84     Solve transport using lp minimizing z ;
85     h(s) = transport.handle; // save instance handle
86   );
87 repeat
88   display$readcollect(h) 'Waiting for next instance to complete';
89   loop(s$handlecollect(h(s)),
90     display$handledelete(h(s)) 'trouble deleting handles';
91     h(s) = 0; // indicate that we have loaded the solution
92   );
93 until card(h) = 0 or timeelapsed > 180; // wait until all models are loaded
94 time(sl) = (jnow-tmp)*24*60*60;
95);
96 display time;

---
96 PARAMETER time time for 100 scenarios
aSyncGrid  4.259,  aSyncThreads  0.496
SolveLink Option – Asynchronous Solves

- Helpful, if large ratio of solver time / GAMS time

```gams
7_dicex_solvellink.gms 7_dicex_solvellink.lst

95 96 * SEQUENTIAL SOLVE
97 loop(seq(s1),
98   tmp = jnow;
99   dice2.solvelink=s1_val(s1);
100  loop(s,
101     solve dice2 using mip maximizing wnx;
102   time(s1) = (jnow-tmp)*24*60*60;
104 );
105
106 * Async SOLVE
107 loop(async(s1),
108   tmp = jnow;
109   dice2.solvelink=s1_val(s1);
110  loop(s,
111     solve dice2 using mip maximizing wnx;
112     h(s) = dice2.handle;   // save instance handle
113   );
114 repeat
115   display$readycollect(h) 'Waiting for next instance to complete';
116   loop(s$handlecollect(h(s)),
117     display$handledelete(h(s)) 'trouble deleting handles';
118     h(s) = 0;   // indicate that we have loaded the solution
119   );
120 until card(h) = 0 or timeelapsed > 180;  // wait until all models are loaded
121   time(s1) = (jnow-tmp)*24*60*60;
122 );
123 option time:3:0:1;
124 display time;
```

ChainScript       29.807
CallModule        30.004
LoadLibrary       28.864
aSyncGrid         9.112
aSyncThreads      7.901
GUSS – Gather-Update-Solve-Scatter
aka Scenario Solver

Generates model once and updates the algebraic model *keeping the model “hot”* inside the solver.
GUSS – Gather-Update-Solve-Scatter
aka Scenario Solver

```plaintext
69 parameter dd(s,i,j) distance by scenario
70 ff(s) freight cost by scenario
71 time(*) time for 100 scenarios;
72 scalar tmp;
73
74 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
75 ff(s) = uniform(0.9,1.1)*f;
76 option limrow=0, limcol=0, solprint=off;
77
78 * GUSS
79 transport.solveLink = 0;
80 tmp = jnow;
81 Set mattrib / system.GUSSModelAttributes /;
82 Parameter
83 xxGUSS(s,i,j) collector for level of x
84 srep(s,matttrib) model attributes like modelstat etc
85 o(*) GUSS options / SkipBaseCase 1 /
86
87 Set dict / s . scenario.''
88 o . opt . srep
89 d . param . dd
90 f . param . ff
91 x . level . xxGUSS /
92
93 Solve transport using lp minimizing z scenario dict;
94 time('GUSS') = (jnow-3mp)*24*60*60;
95
96 display time;
```
Grid & GUSS – Examples from the model library

- **trnsgrid**: [https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_trnsgrid.html](https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_trnsgrid.html)
  - Simple asynchronous solves in a loop, separate collection loop

- **tgridmix**: [https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_tgridmix.html](https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_tgridmix.html)
  - Asynchronous solves in combined submission & collection loop. Keep number of submitted models \(\leq \#\text{threads}\)

- **gussgrid**: [https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_gussgrid.html](https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_gussgrid.html)
  - Asynchronous GUSS-solves in combined submission & collection loop. Keep number of submitted models \(\leq \#\text{threads}\)
A Model is a Model is a Model*

* Freely adapted from the poetry of Gertrude Stein, 1874-1946, American writer
What is a Model?
What is a Model?

- A mathematical model is a description of a system using mathematical language.

➤ Mathematical Programming (MP) Model:
List of Equations
What is a **Model in GAMS**?

- **A mathematical model is a description of a system using mathematical language** (from: Wikipedia)

  ➔ Mathematical Programming (MP) Model:
  List of Equations

- Collection of several intertwined MP Models
  - Data Preparation
  - Data Calibration
  - “Solution” Module (e.g. sequential, parallel loop)
  - Report Module
Model as Communication Vehicle

- Defining scope of a (part of a) project/model
- IT, analysts, managers, model builders have different views
- Misunderstandings common with verbal descriptions
- Use a model to define the scope
- Requirements for such a model
  - Rapid prototyping (max. 1-2 man days)
  - Standard IO interface (Excel)
  - Remote execution (e.g. through Web UI)
Model as **Analytic Framework**

- Optimization models do not allow for any type of vagueness
  - Input data requirements
  - Objectives and constraints
  - Results
- Misunderstandings result in failure of the model
  - Compilation/execution errors
  - Infeasible/unbounded MP models
- Model as a contract
Model as **Contract**

- Good models do not rely on contract (input data)
- Input Module (handles bad data)
  - Simple error checks
  - Analyzing and reporting complex data problems
- Good models (modeling systems) provide access to results via independent *result analyzers* for non model experts
- Analytic framework help define *result metric*
  - e.g. violations of soft constraints
Model as **Cost Saver**

- Most convincing and obvious reason for using an optimization model
- *Science of better (INFORMS)*
- Often exaggerated/difficult to estimate

Model Roles over **Time**

- Communication Vehicle
- Analytic Framework
- Contract
- Cost Saver
Outlook: Parallelization on HPC Systems

From simple sequential to highly parallel solve statements
Off-the-shelf software (speedup via parallelization)

How does GAMS support problem-specific solution approaches that are well suited/customized for HPC?
1. Decompose

2. Use structure exploiting solver

GAMS supports different levels of parallelization

GAMS provides annotation facilities and HPC solver links

Non-zero Plot of matrix

User Annotation to define block structure

Solve sequence of problems (e.g. in Benders Decomp.)

Parallelize
Sequential Solve Statements in Loops

- loop body code in sequence, often with an expensive solve statement:

```plaintext
... // preparatory work
loop(scen,
    ... // setup model
    option clear=s; s(scen) = yes;
    solve mymodel min obj using minlp;
    ... // store results
);
... // reporting
```
Parallel Solves – GAMS Grid Facility

- SolveLink option specifies the solver linking conventions
- Split loop in submission & collection loop:

  ```gams
  ... // preparatory work
  parameter h(scen); mymodel.solvelink=%solvelink.async...%
  loop(scen,
    ... // setup model
    option clear=s; s(scen) = yes;
    solve mymodel min obj using minlp;
    h(scen) = mymodel.handle;
  );
  repeat
    loop(scen$handlecollect(h(scen)));
    ... // store results
    h(scen) = 0;
  );
  until card(h)=0;
  ... // reporting
  ```

- Model generation and loop body code in sequence
- Either file based IO or limited to shared memory
Excursus 1: Message Passing Interface (MPI)

```
mpiexec -n 5 gams myfile
  gams myfile
  gams myfile
  gams myfile
  gams myfile
```
Excursus 1: Message Passing Interface (MPI)

mpiexec -n 5 gams myfile

gams myfile <-- %sysenv.PMI_RANK%=0

Excursus 1: Message Passing Interface (MPI)

myfile.gms – pseudo code

$ifthen $sysEnv.PMIRANK%==0
... // preparatory work
$endif
abort$broadcast(db,0) 'problem with broadcast';
... // setup model
s(scen) = ord(scen)-1$sysEnv.PMIRANK%;
solve mymodel min obj using minlp;
... // store results
abort$gather(db,0) 'problem with gather';
);
$ifthen $sysEnv.PMIRANK%==0
... // reporting
$endif

gather(db,0)

broadcast(db,0)
Excursus 1: Message Passing Interface (MPI)

mpiexec -n 5 gams myfile

myfile.gms – pseudo code

```gams
$ifthen $sysEnv.PMIRANK%==0
... // preparatory work
$endif
abort$broadcast(db,0) 'problem with broadcast';
... // setup model
s(scen) = ord(scen)-1=$sysEnv.PMIRANK%;
solve mymodel min obj using minlp;
... // store results
abort$gather(db,0) 'problem with gather';
);
$ifthen $sysEnv.PMIRANK%==0
... // reporting
$endif
```

gams myfile <-- %sysenv.PMI_RANK%==0

gams myfile <-- %sysenv.PMI_RANK%==1

gams myfile <-- %sysenv.PMI_RANK%==2

gams myfile <-- %sysenv.PMI_RANK%==3

gams myfile <-- %sysenv.PMI_RANK%==4

MG | MG | MG
---|----|----
S  | S  | S

...
Excursus 1: Message Passing Interface (MPI)

```
mpiexec -n 5 gams myfile
```

myfile.gms – pseudo code
```
$ifthen $sysEnv.PMIRANK%==0
... // preparatory work
$endif
abort$broadcast(db,0) 'problem with broadcast';
... // setup model
s(scen) = ord(scen)-1=$sysEnv.PMIRANK%;
solve mymodel min obj using minlp;
... // store results
abort$gather(db,0) 'problem with gather';
);
$ifthen $sysEnv.PMIRANK%==0
... // reporting
$endif
```

- Requires reorganization of the code but allows parallel solve
- Distribute/merging data easy (part of MPI)
- Network based communication
- Need to make GAMS aware of MPI → Embedded Code
Excursus 2: Embedded Code Facility

24.9.1 Major release (August 30, 2017)

Acknowledgments

We would like to thank all of our users who have reported problems and made suggestions for improving this release. In particular, we thank Etienne Ayotte-Sauvé, Wolfgang Britz, Florian Habermacher, Florian Häberlein, Maximilian Held, Ignacio Herrero, Hanspeter Höschle, Erwin Kalvelagen, Toni Lastusilta, John Ross, and Tom

GAMS System

GAMS

- New feature, the **Embedded Code Facility**: This extends the connectivity of GAMS to other programming languages. It allows the use of Python code during compile and execution time. **GAMS symbols are shared with the external code, so no communication via disk is necessary.**

  The embedded code feature is available on Linux, MacOS X, and Windows. For these platforms, a Python 3.6 installation is included with the GAMS distribution. If the user wants to work with a different Python 3.6, installed separately, for models with embedded code the new command line option `pySetup` needs to be set to 0.

  **Note**

  This feature is currently in beta status. Any feedback to support@gams.com is appreciated.

- New command line option `procDirPath`: Specifies the directory where the process directory should be created.
Example: Sequential Benders Decomposition

```plaintext
set scen   'scenario set' / scen1*scen100 /
  s(scen) 'dynamic scenario subset'
k      'benders iterations' / k1*k1000 /;

... // preparatory work

loop(k$( NOT done ),
    ...
    // setup model for master-problem
    solve master min obj_master use lp;
    ...
    // fix first stage variables

loop(scen,
    ...
    // setup model for sub-problem
    option clear=s; s(scen) = yes;
    solve sub min obj_sub use lp;
    ...
    // process results
);

... // compute cuts for next master
...
... // free fixed first stage variables
...
... // set done=1 if convergence criterion is met
)

... // reporting
```
## Example: Parallel Benders with mpi4py

<table>
<thead>
<tr>
<th>PMI_RANK=0</th>
<th>PMI_RANK&gt;=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>set scen</strong> 'scenario set' / scen1*scen100 /</td>
<td><strong>set scen</strong> 'scenario set' / scen1*scen100 /</td>
</tr>
<tr>
<td>s(scen) 'dynamic scenario subset'</td>
<td>s(scen) 'dynamic scenario subset'</td>
</tr>
<tr>
<td>k 'benders iterations' / k1*k1000 ;</td>
<td>k 'benders iterations' / k1*k1000 ;</td>
</tr>
</tbody>
</table>

... embeddedCode Python:
```python
from mpi4py import *
comm = MPI.COMM_WORLD
```

... // preparatory work

```python
ifthen.MPI
```

```python
sysenv.PMI_RANK %
```

```python
loop(k$ ( NOT done ),
... // setup model for master-problem
solve master min obj_master use lp;
... // fix first stage variables
```

```python
continueEmbeddedCode:
```
```python
comm.bcast([[done]] + <data for sub>, root=0)
```
```python
cut = comm.gather(None, root=0)[1:]
```
```python
... // gathered data \rightarrow GAMS data struct.
```
```python
pauseEmbeddedCode <load GAMS data struct.>
```
```python
... // compute cuts
```
```python
... // free fixed first stage variables
```
```python
... // set done=1 if convergence criterion is met
```

```python
continueEmbeddedCode:
```
```python
comm.bcast([[done],<empty>], root=0)
```
```python
endEmbeddedCode
```

... // reporting

```python
$else.MPI
```
```python
s(scen) = ord(scen)%sysenv.PMI_RANK%;
```
```python
while(1,
```
```python
continueEmbeddedCode:
```
```python
primal_solution = comm.bcast(_None, root=0)
```
```python
// broadcasted data \rightarrow GAMS data struct.
```
```python
pauseEmbeddedCode <GAMS data struct.>
```
```python
abort.noerror$done 'terminating subprocess';
```
```python
solve sub min obj_sub use lp;
```
```python
... // process results
```
```python
continueEmbeddedCode:
```
```python
comm.gather(<subproblem results>), root=0 )
```
```python
pauseEmbeddedCode
```

```python
);
```
```python
$endif.MPI
```
Computational Result(s)

- Two-stage stochastic problem emerged from energy system model
- 100 scenarios
- Deterministic Equivalent:
  - 21,029,101 rows, 23,217,077 columns, 85,721,477 non-zeroes
- Benders:
  - Master: up to 553 rows, 177 columns, 24,911 non-zeroes
  - Sub: 210,282 rows 232,161 columns 696,461 non-zeroes
  - 19 lines of Python Code + some refactorization of GAMS code for MPI version

<table>
<thead>
<tr>
<th>Method</th>
<th>sub-problems</th>
<th>master-problem</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic Equivalent¹</td>
<td></td>
<td></td>
<td>4059.00</td>
</tr>
<tr>
<td>Seq. Benders²</td>
<td>2394.92</td>
<td>0.18</td>
<td>2395.10</td>
</tr>
<tr>
<td>MPI Benders³</td>
<td>28.35</td>
<td>0.16</td>
<td>28.51</td>
</tr>
</tbody>
</table>

All runs were made with GAMS 25.1.2 on JURECA@JSC with 24 cores per node, 2.5 GHz, (Intel Xeon E5-2680 v3 Haswell), 128 GB RAM

1: single node, 16 cores, CPLEX barrier, no crossover
2: single node, 4 cores per solve statement, CPLEX barrier, advind 0
3: 17 nodes, 404 cores in total, 4 cores per solve statement, CPLEX barrier, advind 0
Deployment of GAMS models
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
Low level APIs $\rightarrow$ Object Oriented API

- Low level APIs
  - GDX, OPT, GAMSX, GMO, ...
  - High performance and flexibility
  - Automatically generated imperative APIs for several languages (C, C++, C#, Delphi, Java, Python, VBA, ...)

- Object Oriented GAMS API
  - Additional layer on top of the low level APIs
  - Object Oriented
  - Written by hand to meet the specific requirements of different Object Oriented languages
• GAMS comes with several OO APIs (Python, Java, C++, C#, ...) to develop applications

→ Programming required to build your applications
From GAMS Model to Visual Web User Interface

Currently under Development
From GAMS Model to Visual Web User Interface

Basic setup:
✓ Annotating GAMS model (defining the input and output data to be displayed in the WebUI)
Basic Setup – GAMS Model Annotations

- Fully functional interface by only specifying input and output data
- Tabular view of input (editable) and output data
- Graphical visualization via pivot charts
Basic Setup –
GAMS Model Annotations

```gams
$onExternalInput
Set date 'date'
  symbol 'stockSymbol';

Parameter
  stockData(date,symbol,hrd)  'data of stock on date ### { "headers": {"date": {"readonly":true} } }';

Scalar
  maxstock  'maximum number of stocks to select ### { "slider": {"min":1, "max":card(stockdata)} }';
  trainingdays 'number of days for training ### { "slider": {"min":1, "max":card(stockdata)} }';
$offExternalInput

$offExternalOutput
Set wHdri 'w header' / 'weight' /
  fHdr 'fund header' / 'dow jones','index fund' /
  errHdr 'stock symbol header' / 'absolute error train','absolute error test' /;

Parameter
  partOfPortfolio(symbol,wHdri)  'what part of the portfolio'
  dowVSindex(date,fHdr)  'dow jones vs. index fund'
  absererror(date,errHdr)  'absolute error'
Singleton Set lastDayTraining(date)  'last date of training period ### vertical marker in chart' ;
$offExternalOutput

$if not exist webui.gms
$if set GMSWEBUI $abort Asked to do webui but can't find webui.gms. Set idir=-path/to/webui
$batinclude webui
```
Basic Setup –
GAMS Model Annotations

```gams
$onExternalInput
Set date /date 1, date 2, date 3/;
symbol 'stockSymbol';

Parameter
stockData(date,symbol,hrd) 'data of stock on date ### { "headers":{"date":{"readonly":true}} }';

Scalar
maxstock 'maximum number of stocks to select ### { "slider":{"min":1, "max":card(stockdata$}

errHdrt 'number of days for training ### { "slider":{"min":1, "max":card(stockdata$}

Set
fHdr 'fund header' /'weight' /
errHdrt 'index fund' /'dow jones','index fund' /
errHdr 'stock symbol header' /'absolute error train', 'absolute error test' /;

Parameter
partOfPortfolio(symbol,wHdrt) 'what part of the portfolio'
dowVSindex(date,fHdr) 'dow jones vs. index fund'
absererror(date,errHdr) 'absolute error'

Singleton Set
lastDayTraining(date) 'last date of training period ### vertical marker in chart';
```

$if not exist webui.gms $abort Asked to do webui but can't find webui.gms. Set idir=path/to/webui
$batinclude webui
Basic Setup – GAMS Model Annotations

```gams
$OnExternalInput
Set date 'date'
  symbol 'stockSymbol';

Parameter
  stockData(date,symbol,hrd) 'data of stock on date ### { "headers": { "date": { "readonly": true } } }';
  maxstock 'maximum number of stocks to select ### { "slider": { "min": 1, "max": card(stockdata$) } }';
  trainingdays 'number of days for training ### { "slider": { "min": 1, "max": card(stockdata$) } }';
$OffExternalInput

$OnExternalOutput
Set wHdr 'w header' / 'weight' /
  fHdr 'fund header' / 'dow jones', 'index fund' /
  errHdr 'stock symbol header' / 'absolute error train', 'absolute error test' /
Parameter
  partOfPortfolio(symbol,wHdr) 'what part of the portfolio'
  dowVSindex(date,fHdr) 'dow jones vs. index fund'
  abserror(date,errHdr) 'absolute error'
Singleton Set lastDayTraining(date) 'last date of training period ### vertical marker in chart';
$OffExternalOutput

@if not exist webui.gms
  $OnInclude
  Set GMSScript /
  $batinclude webui
```
From GAMS Model to Visual Web User Interface

1. Basic setup:
   ✓ Annotating GAMS model (defining the input and output data to be displayed in the WebUI)

2. Advanced setup:
   ✓ Configuration of standard graphics and UI
   ✓ Sophisticated (custom) graphics (R API)
   ✓ Scenario management with internal database
Advanced Setup – Configuration

- Configuration via JSON file
- Access to a number of pre-implemented tools for graphical representation
- Focus on configuration, not programming

```json
"dow Jones": {
  "label": "Dow Jones index",
  "fillGraph": true
},
"index fund": {
  "label": "Index fund"
}
"outType": "graph",
"tool": "dygraph",
"title": "Dow Jones vs. Index Fund",
"xdata": [
  "Apr 2016", "Jul 2016", "Oct 2016"
]
"ydata":
  [100, 105, 110, 115, 120]
```

Dow Jones vs. Index Fund

- Dow Jones index
- Index fund
Advanced Setup – Sophisticated graphics

- Sophisticated graphics created in R can be included as modules
- Access to the entire R ecosystem
- Easily extendable with the wide spectrum of the R programming language
From GAMS Model to Visual Web User Interface

Initialization:
✓ Annotating GAMS model (defining the input and output data to be displayed in the WebUI)

Advanced setup:
✓ Configuration of standard graphics and UI
✓ Sophisticated (custom) graphics (R API)
✓ Scenario management with internal database

Enterprise setup:
✓ User- and Application management
Enterprise Setup – User and Application Management

- Local or server-based solution
- User authentication (e.g. LDAP, OAuth 2.0, Google, GitHub, Facebook)
- Multi-Application support with docker-based technology

Yaml config file:
```yaml
- name: simple
display-name: Simple
logo-url: file://localhost/home/ec2-user/simple.png
description: A Simplified Energy System Model Annotation
docker-cmd: ["R", ",", "shiny::runApp(\'/root/GMSWebJTI\')"]
docker-image: gams/app
docker-network: "my-network"
docker-env:
  SHINYPROXY_MODELNAME: simple
groups: GAMS_team, guest
```
Enhanced Model Deployment in GAMS using R/Shiny

• Application connects Web User Interface with a GAMS model

• User Interface allows
  ✓ Data exchange via local files or database
  ✓ Modification of the input data
  ✓ Extensive visualization options
  ✓ Comparison of different scenarios
  ✓ Multi-user support based on Docker technology
  ✓ User authentication

• Tool with intuitive interface for planners (configuration vs. programming)

• This “product” is currently under development. If you are interested in getting involved, please contact support@gams.com
Welcome to Jupyter @ GAMS!

Enter your credentials in order to sign in or contact GAMS Support for further information.

Getting Started

- Introduction
- Milco Example
- PickStock Example
- A GAMS Tutorial by Richard E. Rosenthal

Further Help

- Jupyter Notebook Users Manual (from Bryn Mawr College)
- GAMS World Forum
- Contact GAMS

Currently under Development
GAMS Jupyter Example

In [17]:
```python
# GAMS
Parameter fund(date) 'Index fund report parameter'; fund(d) = sum(s, price(d, s)*w.l(s));
Parameter error(date) 'Absolute error'; error(d) = abs(index(d)-fund(d));
```

Plotting of the results

In [18]:
```python
# GAMS pull
fig, ax = plt.subplots()
index.plot(y="value", ax=ax, xticks=[0, trainingDays, len(date)], yticks=[], label="Dow Jones")
fund.plot(y="value", ax=ax, xticks=[0, trainingDays, len(date)], yticks=[], label="Index Fund")
```

![Graph showing comparison between Dow Jones and Index Fund]
Using GAMS Jupyter Notebooks to tell “optimization stories”

• Runs in a browser/on a server → No local installation needed
• Allows to use notebook technology in combination with GAMS
• Notebooks allow to combine GAMS and Python
  • GAMS works great with well structured data and optimization models
  • Python is very rich in features to retrieve, manipulate, and visualize data that comes in all sort of ways
  • Combining GAMS and Python in a notebook makes it easy to tell optimization story with text, data, graphs, math, and models

• This “product” is currently under development. Give it a try at https://jupyterhub.gams.com/hub/login
Outlook: Stochastic Programming
Stochastic Programming in GAMS

**EMP/SP**
- Simple interface to add uncertainty to existing deterministic models
- (EMP) Keywords to describe uncertainty include: discrete and parametric random variables, stages, chance constraints, Value at Risk, ...
- Available solution methods:
  - Automatic generation of Deterministic Equivalent (can be solved with any solver)
  - Specialized commercial algorithms (DECIS, LINDO)
Transport Example – Uncertain Demand

Decisions to make

➢ First-stage decision: How many units should be shipped “here and now” (without knowing the outcome)
➢ Second-stage (recourse) decision:
  ➢ How can the model react if we do not ship enough?
  ➢ Penalties for “bad” first-stage decisions, e.g. buy additional cases $u(j)$ at the demand location:

\[
\text{cost} \quad \text{z} = \text{e} = \sum_{(i,j)} c(i,j) \times x(i,j) + \sum_j 0.3 \times u(j) ;
\]

\[
\text{demand} \quad \text{sp}(j) \quad \text{sum}_i \times x(i,j) = \text{g} = bf \times b(j) - u(j) ;
\]
Uncertain Demand: GAMS Algebra

GAMS Studio

110

Uncertain Demand: GAMS Algebra

23 * Stochastic Program with uncertain demand
24 Positive variable u(j) unsatisfied demand:
25 Scalar  bf demand factor / 1 /;
26 Equation costsp  define objective function for SP
27    demandsp(j) demand satisfaction in SP:
28
29 costsp..  z =e= sum((i,j), c(i,j)*x(i,j)) + sum(j, 0.3*u(j));
30 demandsp(j).. sum(i, x(i,j)) >= bf*b(j) - u(j):
31
32 Model transportSP / costsp, demandsp, supply /:
33 File emp / 'emp.info' /; put emp:
34 $input
35 randvar bf discrete 0.3 0.9
36         0.5 1.0
37         0.2 1.1
38 stage 2 bf u demandsp
39 $offput
40 Putclose emp;
41
42 Set scen scenarios / s1*s4 /;
43 Parameter
44 s_bf(scen)  demand factor for realization by scenario
45 s_x(scen,i,j) shipment per scenario
46 s_u(scen,j) unsatisfied demand per scenario (bought cases):
47
48 Set dict / scen . scenario . ''
49    bf . randvar . s_bf
50    x . level . s_x
51    u . level . s_u /;
52
53 option emp=lindo;
54 Solve transportSP min z use emp scenario dict;
Uncertain Demand: Results
The Extended Mathematical Programming (EMP) framework is used to replace parameters in the model by random variables.

Support for Multi-stage recourse problems and chance constraint models.

Easy to add uncertainty to existing deterministic models, to either use specialized algorithms or create Deterministic Equivalent (new free solver DE).

More information: [https://www.gams.com/latest/docs/UG_EMP_SP.html](https://www.gams.com/latest/docs/UG_EMP_SP.html)