GAMS – An Introduction

Steven Dirkse & Lutz Westermann
GAMS Development Corp. / GAMS Software GmbH
Agenda

GAMS at a Glance

GAMS – Hands On Examples

Outlook

• Deployment of GAMS Models
  • APIs – Application Programming Interfaces to GAMS
  • Low Level APIs
  • Object Oriented APIs
  • Using R/Shiny to deploy and visualize GAMS models in a Web Interface
GAMS at a Glance
The Vision

GAMS came into being!

1976 - A World Bank Slide

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:

- Microsoft Windows
- Linux
- macOS
- Solaris
- AIX

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
  - ...

---

GAMS

GDX

SOLVER
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 – Hands On Examples
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

- SP: Uncertainty
  - E.g. Uncertain Demand
A Simple Transportation Problem

**Canning Plants** (supply)  [Shipments]  **Markets** (demand)

(Number of cases)

Freight: $90 case / thousand miles

- **Seattle** (capacity: 350)
- **San-Diego** (capacity: 600)
- **Chicago** (demand: 300)
- **Topeka** (demand: 275)
- **New-York** (demand: 325)
Minimize Transportation cost
subject to
Demand satisfaction at markets
Supply constraints

A Simple Transportation Problem

Seattle
Supply (cases) 350
Distance (thousand miles) 1.8 1.7 2.5
Demand (cases) 275
Topeka
San Diego
Supply (cases) 600
Distance (thousand miles) 1.4 2.5 1.8
Demand (cases) 300 325
Chicago
New York

Freight: $90 case / thousand miles

Supply
Demand
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)

\[
\begin{align*}
\min & \quad \sum_i \sum_j c_{ij} \cdot x_{ij} & \text{(Minimize total transportation cost)} \\
\text{subject to} & \quad \sum_j x_{ij} \leq a_i & \quad \forall \; i \quad \text{(Shipments from each plant \leq supply capacity)} \\
& \quad \sum_i x_{ij} \geq b_j & \quad \forall \; j \quad \text{(Shipments to each market \geq demand)} \\
& \quad x_{ij} \geq 0 & \quad \forall i, j \quad \text{(Do not ship from market to plant)} \\
& \quad i, j \in \mathbb{N} & 
\end{align*}
\]
GAMS Syntax (LP Model)

Variables
\[ x(i,j) \text{ shipment quantities in cases} \]
\[ z \text{ total transportation costs in thousands of dollars} \]

Positive Variable \( x \);

Equations
\[ \text{cost} \quad \text{define objective function} \]
\[ \text{supply}(i) \quad \text{observe supply limit at plant } i \]
\[ \text{demand}(j) \quad \text{satisfy demand at market } j \]

\[ \text{cost ..} \]
\[ z = e = \text{sum(}(i,j), c(i,j)*x(i,j)) \]

\[ \text{supply}(i) .. \]
\[ \text{sum}(j, x(i,j)) = l = a(i) \]

\[ \text{demand}(j) .. \]
\[ \text{sum}(i, x(i,j)) = g = b(j) \]

Model transportLP /all/ ;

Solve transportLP using lp minimizing z ;
GAMS Syntax (Data Input)

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 gdxrwx data.xlsx o=data.gdx @instructions.txt
$if errorlevel 1 $abort Error preparing data
$gdxin data.gdx
$load i<d1 j<d2 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)

San-Diego (capacity: 600) → Topeka (demand: 275)

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

SP
- Uncertainty
- E.g. Uncertain Demand
MIP Model: Minimum Shipment of 100 cases

- Shipment volume: $x$ (continuous variable)
- Discrete decision: ship (binary variable)

**Add constraints:**

\[ x_{i,j} \geq 100 \cdot \text{ship}_{i,j} \quad \forall i,j \quad \text{(if ship=1, then ship at least 100)} \]
\[ x_{i,j} \leq \text{bigM} \cdot \text{ship}_{i,j} \quad \forall i,j \quad \text{(if ship=0, then do not ship at all)} \]
\[ \text{ship}_{i,j} \in \{0,1\} \]
MIP Model: GAMS Syntax

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

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

SP
• Uncertainty
• E.g. Uncertain Demand
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
\[ \min \sum_i \sum_j c_{ij} \cdot \beta \cdot x_{ij} \]  
(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

<table>
<thead>
<tr>
<th>Entry</th>
<th>Name</th>
<th>Type</th>
<th>Dim</th>
<th>Recs</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>300</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>Parameter</td>
<td>6</td>
<td></td>
<td>275</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>e</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>f</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>g</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>h</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>i</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>j</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>k</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>l</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>m</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>n</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>o</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>p</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>q</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>s</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>t</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>u</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>v</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>w</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>x</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>y</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>z</td>
<td>Parameter</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

### Column List

- seattle
- san-diego
- new-york
- chicago
- topeka

### Execution Time

0.0000 seconds
MINLP Model: Solution

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

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

New-York (demand: 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

SP
- Uncertainty
- E.g. Uncertain Demand
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.

```gams
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
  • Hint: Look, e.g., at the GAMS function \texttt{uniform}

• 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 \texttt{jNow}
SolveLink Option – Sequential Solves

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

---
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/;

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 scalar h(s) scenario handle;
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 * Async SOLVE
78 loop(sl,
79 tmp = jnow;
80 transport.solvelink=sl_val(sl);
81 loop(s,
82 d(i,j) = dd(s,i,j);
83 Solve transport using lp minimizing z;
84 h(s) = transport.handle; // save instance handle
85);
86 repeat
87 display$readcollect(h) 'Waiting for next instance to complete';
88 loop(s$handlecollect(h(s))
89 display$handledelete(h(s)) 'trouble deleting handles';
90 h(s) = 0; // indicate that we have loaded the solution
91 );
92 until card(h) = 0 or timeelapsed > 180; // wait until all models are loaded
94 time(sl) = (jnow-temp)*24*60*60;
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
95 * SEQUENTIAL SOLVE
96 loop(seq(sl),
97     tmp = jnow;
98     dice2.solveLink = sl_val(sl);
99     loop(s,
100         solve dice2 using mip maximizing wnx;
101     );
102     time(sl) = (jnow-tmp)*24*60*60;
103 );
104
105 * Async SOLVE
106 loop(async(sl),
107     tmp = jnow;
108     dice2.solveLink = sl_val(sl);
109     loop(s,
110         solve dice2 using mip maximizing wnx;
111         h(s) = dice2.handle;    // save instance handle
112     );
113 repeat
114     display$readyCollect(h) 'Waiting for next instance to complete';
115     loop(s$handleCollect(h(s)),
116         display$handleDelete(h(s)) 'trouble deleting handles';
117         h(s) = 0;    // indicate that we have loaded the solution
118     );
119     until card(h) = 0 or timeelapsed > 180; // wait until all models are loaded
120     time(sl) = (jnow-tmp)*24*60*60;
121 );
122 option time:3:0:1;
123 display time;
```
GUSS – Gather-Update-Solve-Scatter
aka Scenario Solver

Simple Serial Solve Loop

Scenario Solver/GUSS

Generation

Solution

Update

Loop

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

```
8_transport_GUSS_solveld.gms  8_transport_GUSS_solveld.lst

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.solveld = 0;
80 tmp = jnow;
81 Set mattrb / system.GUSSModelAttributes /
82 Parameter
83     xxGUSS(s,i,j) collector for level of x
84     srep(s, mattrb) model attributes like modelstat etc
85     o(*) GUSS options / SkipBaseCase 1 /
86
87 Set dxt / 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-tmp)*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 <= #threads

• guassgrid: [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 <= #threads
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

SP
• Uncertainty
• E.g. Uncertain Demand
**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

**Uncertain demand factor** $bf$

### 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:

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

  \[
  \text{demandsp}(j): \quad \sum_i x(i,j) \geq bf b(j) - u(j);
  \]
Uncertain Demand: GAMS Algebra

A Stochastic Program with uncertain demand
Positive variable \( u(j) \) unsatisfied demand:
Scalar \( \text{bf} \) demand factor / 1 /;
Equation costsp define objective function for SP
demandsp(j) demand satisfaction in SP;

\[
z = \sum (i, j), c(i, j) * x(i, j)) + \sum (j, 0.3 * u(j));
\]
demandsp(j) = \sum (i, x(i, j)) = \text{bf} * b(j) - u(j);

Model transportSP / costsp, demandsp, supply /;
File emp / '{emp.info}' / put emp;
$computation
randvar \text{bf} \text{ discrete} 0.3 \quad 0.9
0.5 \quad 1.0
0.2 \quad 1.1
end stage 2 \text{bf} \text{ u demandsp}
end computation
Putclose emp;
Set scen scenarios / s1*s4 /
Parameter
\( s_{bf}(\text{scen}) \) demand factor for realization by scenario
\( s_{x}(\text{scen}, i, j) \) shipment per scenario
\( s_{u}(\text{scen}, j) \) unsatisfied demand per scenario (bought cases):

\[
\text{Set dict / scen . scenario . ''
\text{bf . randvar . s_{bf}}
\text{x . level . s_{x}}
\text{u . level . s_{u}} /
\]
Option emp=lindo;
Solve transportSP min z use emp scenario dict;

Hands-On
Uncertain Demand: Results

---

GAMS Studio

1. **5_transp_LP_MIP_MINLP_SP**
   - **5_transp_LP_MIP_MINLP_SP.gdx**
   - **5_transp_LP_MIP_MINLP_SP.gdx**
   - **5_transp_LP_MIP_MINLP_SP.gams**
   - **5_transp_LP_MIP_MINLP_SP.gms**

2. **Columns**
   - **x**
   - **s**
   - **ship**

3. **Model Statistics**
   - **SOLVE transportLPMinLp...**

4. **Solution Report**
   - **SOLVE transportLPMinLp...**

5. **Solve Equation List**
   - **SOLVE transportLPMinLp...**

6. **Column Listing**
   - **SOLVE transportLPMinLp...**

---

1. **1031**
   - **156 PARAMETER s_bf demand factor for realization by scenario**

2. **1032**
   - **s1 0.900, s2 1.000, s3 1.100**

3. **1033**
   - **156 PARAMETER s_b demand per scenario**

4. **1034**
   - **new-york, chicago, topeka**

5. **1035**
   - **s1 292.500, 270.000, 247.800**

6. **1036**
   - **s2 325.000, 300.000, 275.000**

7. **1037**
   - **s3 357.500, 330.000, 302.500**

---

1. **1045**
   - **156 PARAMETER s_x shipment per scenario**

2. **1046**
   - **new-york, chicago, topeka**

3. **1047**
   - **s1.seattle 50.000, 300.000, 275.000**

4. **1048**
   - **s2.seattle 50.000, 300.000, 275.000**

5. **1049**
   - **s3.seattle 50.000, 300.000, 275.000**

6. **1050**
   - **s1.san-diego 242.500, 242.500, 242.500**

7. **1051**
   - **s2.san-diego 50.000, 300.000, 275.000**

8. **1052**
   - **s3.san-diego 50.000, 300.000, 275.000**

---

1. **1057**
   - **156 PARAMETER s_u unsatisfied demand per scenario (bought cases)**

2. **1058**
   - **new-york, chicago, topeka**

3. **1059**
   - **s2 32.500, 30.000, 27.500**

4. **1060**
   - **s3 65.000, 30.000, 27.500**

---

1085 lines, 1031/17 RO, UTF-8
Stochastic Programming in GAMS

• 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)
Outlook: Deployment of GAMS Models

- APIs – Application Programming Interfaces to GAMS
  - Low Level APIs
  - Object Oriented APIs
- Using R/Shiny to deploy and visualize GAMS models in a Web Interface
Excel and GAMS

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Distance</strong></td>
<td>New-York</td>
<td>Chicago</td>
<td>Topeka</td>
<td>Supply</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>Seattle</strong></td>
<td>2.5</td>
<td>1.7</td>
<td>1.8</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>San-Diego</strong></td>
<td>2.5</td>
<td>1.8</td>
<td>1.4</td>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Demand</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>Freight cost</strong></td>
<td></td>
<td></td>
<td></td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>SHIPMENT</strong></td>
<td>New-York</td>
<td>Chicago</td>
<td>Topeka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Seattle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>San-Diego</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GAMS Directory: \C:\GAMS\win64\24.8.5\  
Working Directory: \C:\tmp\  
Solver: CPLEX

Clear Solution

SOLVE LP   SOLVE MIP

Hands-On TransXLS
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 → 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
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
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$) } };
  trainingdays  'number of days for training'  
    { "slider": { "min": 1, "max": card(stockdata$) } };

Set wHdr  'w header'  
  fHdr  'fund header'  
  errHdr  'stock symbol header'  
    'dow jones', 'index fund'  
    'absolute error train', 'absolute error test';

Parameter
  partOfPortfolio(symbol,wHdr)  'what part of the portfolio'  
  dowVIndex(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';

$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

```
Set date 'date';
symbol 'stockSymbol';

Parameter
  stockData(date,symbol,hrd) 'data of stock on date'  
    /.../;

Scalar
  maxstock 'maximum number of stocks to select'  
    /.../;

Scalar
  numDaysForTraining 'number of days for training'  
    /.../;

Set
  fHdr 'fund header'  
    /.../;

Set
  errHdr 'stock symbol header'  
    /.../;

Parameter
  partOfPortfolio(symbol,wHdr) 'what part of the portfolio'
  dowVSindex(date,fHd) 'dow jones vs. index fund'
  abserror(date,errHd) 'absolute error'

Singleton Set
    lastDayTraining(date) 'last date of training period'  
    /.../;

Conditional:
  $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

7 $onExternalInput
8 Set date 'date' symbol 'stockSymbol';
9
10 Parameter
11   stockData(date,symbol,hrd) 'data of stock on date' ### {"headers":{"date":{"readonly":true}}} ;
12 maxstock 'maximum number of stocks to select' ### {"slider":{"min":1, "max":card(stockData$)}
13 trainingdays 'number of days for training' ### {"slider":{"min":1, "max":card(stockData$}
14 $offExternalInput
15
16 $onExternalOutput
17 Set wHdr 'w header' / 'weight' /
18   fHdr 'fund header' / 'dow jones','index fund' /
19   errHdr 'stock symbol header' / 'absolute error train','absolute error test' /;
20 Parameter
21   partOfPortfolio(symbol,wHdr) 'what part of the portfolio'
22   dowVIndex(date,fHdr) 'dow jones vs. index fund'
23   aberror(date,errHdr) 'absolute error'
24 Singleton Set lastDayTraining(date) 'last date of training period' ### vertical marker in chart' ;
25 $offExternalOutput
26
27 $if not exist webui.gms
28 $include GAMSWebui
29 $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 (Configuration Generator)
✓ Access to a number of pre-implemented tools for graphical representation
✓ Focus on configuration, not programming
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

1. **Initialization:**
   - ✓ 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

3. **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", "-e", "shiny::runApp('/root/GMSWebUI')"]
docker-image: gams/app
docker-network: "my-network"
docker-env:
  SHINYPROXY_MODELNAME: simple
groups: GAMS_team, guest
```

Hands-On
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
Thank You