



GAMS

An Introduction

Lutz Westermann, GAMS Software GmbH



Company Background



Agenda

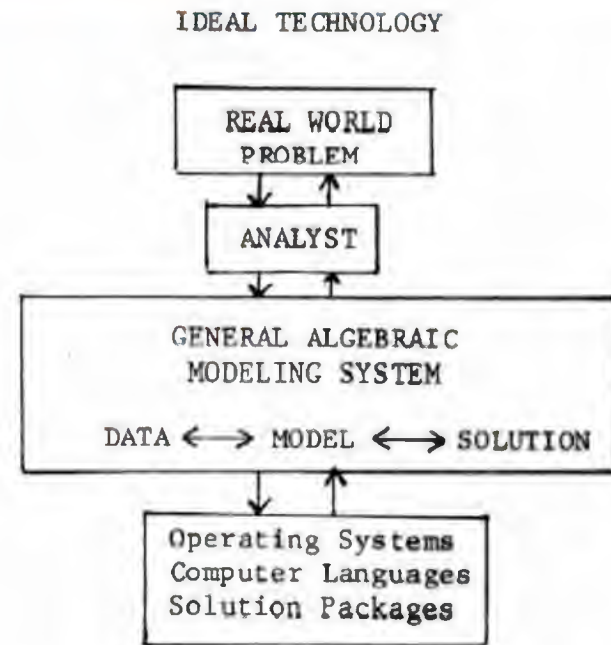
What is GAMS?

- Algebraic Modeling Languages – A Success Story
- GAMS – Highlights and Design Principles
- Striving for Innovation and Compatibility
 - New Modeling and Solution Concepts
 - Software Quality Assurance

A Simple Example

1976 - A World Bank Slide

The
Vision

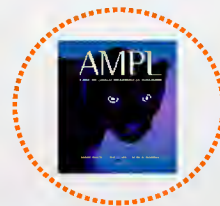


- RESULT:
- Limited drain of resources
 - Same representation of models for humans and machines
 - Model representation is also model documentation

2012 INFORMS **Impact Prize**

36 Years
later

Originators of Algebraic Modeling Languages

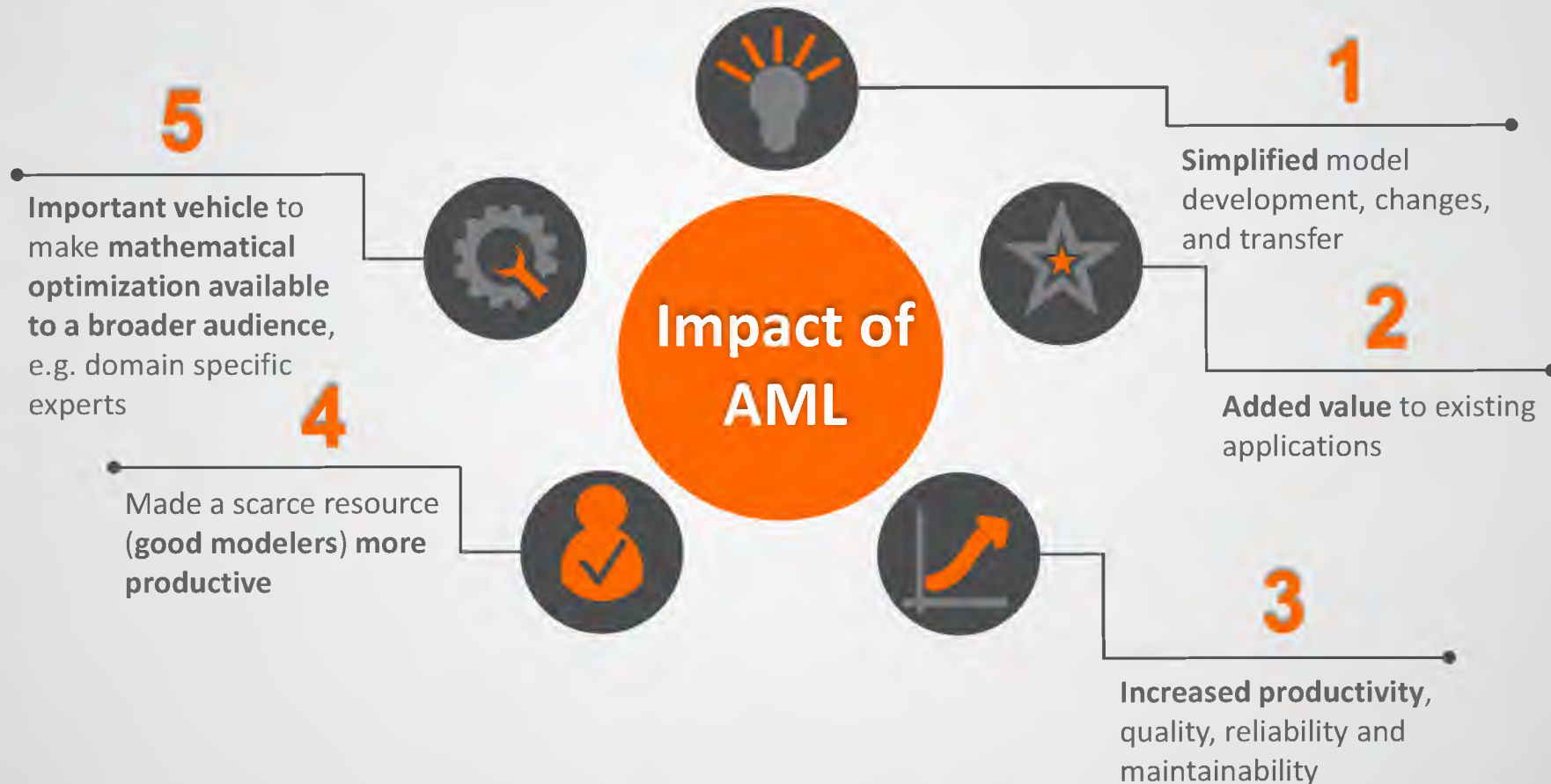


Algebraic Modeling Languages (AML)

- 1 {
 - High-level **computer programming languages**
 - Formulation of **mathematical optimization problems**
 - **Notation similar to algebraic notation**
- 2 {
 - **Do not solve problems directly**, but offer links to state-of-the-art algorithms (“solver-links”)

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

Impact of Algebraic Modeling Languages



What does a modeler **have to think about?**



1. Problem
2. Mathematics
3. Programming
4. Performance
5. Scalability
6. Connectivity
7. Deployment
8. Maintenance (Life Cycle)
9. ...

Why is GAMS a tool for him?

Broad User **Community and Network**

GAMS used in more than 120 countries



25+ Years
GAMS Development

Broad User **Community and Network**

More than 10,000 licenses

50% academic users, 50% commercial users



25+ Years
GAMS Development

6,000+ monthly downloads of the
free system

Broad Range of **Application Areas**

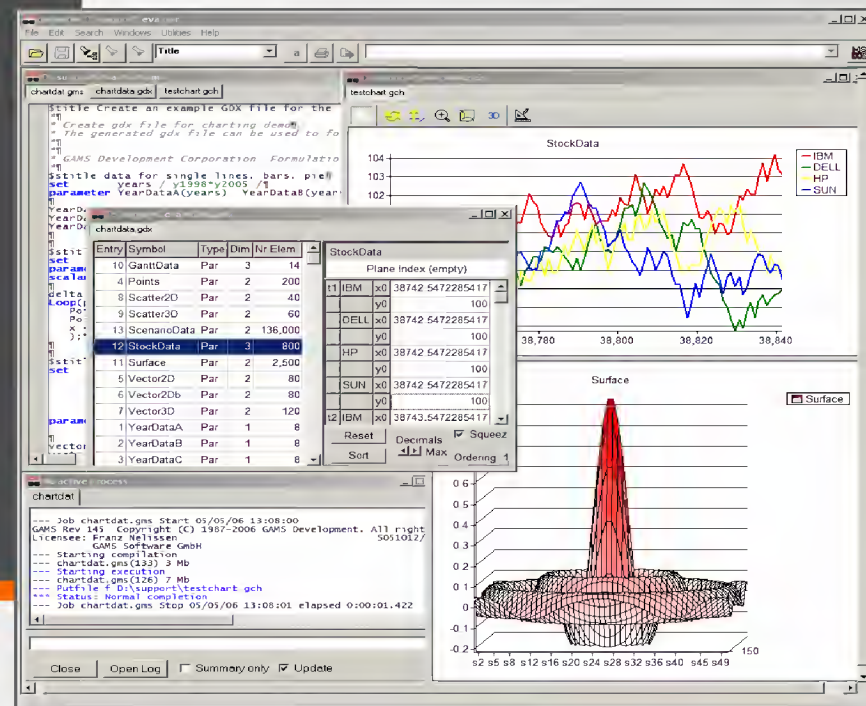
Agricultural Economics	Applied General Equilibrium
Chemical Engineering	Economic Development
Econometrics	Energy
Environmental Economics	Engineering
Finance	Forestry
International Trade	Logistics
Macro Economics	Military
Management Science/OR	Mathematics
Micro Economics	Physics

25+ Years
GAMS Development

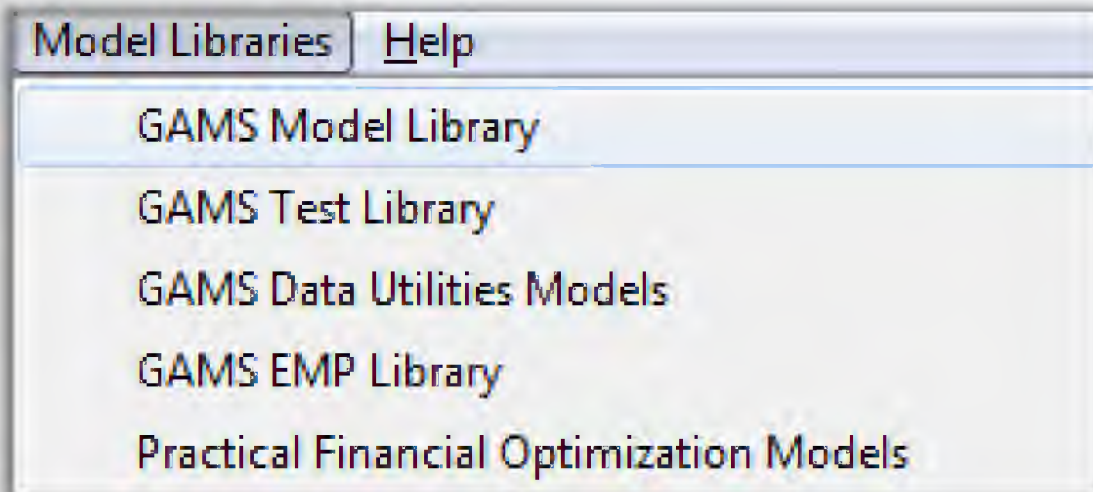
Strong Development Environment

GAMS IDE

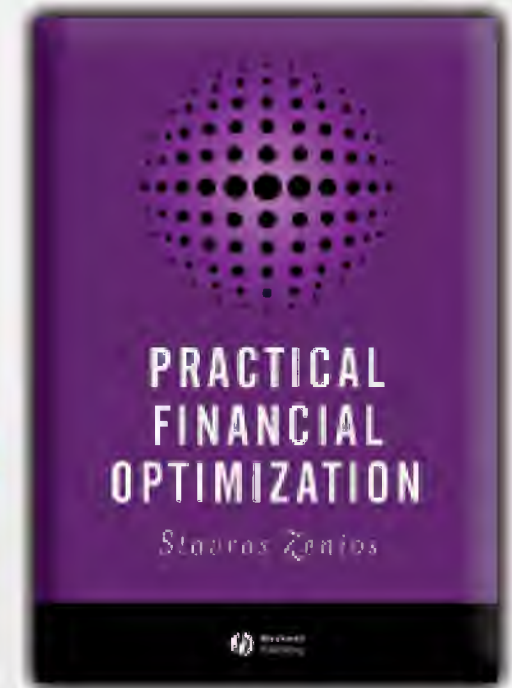
- Project management
- Editor / Syntax coloring / Spell checks
- Listing file / Tree view / Syntax-error navigation
- Model Debugging / Profiling
- Solver selection / Option selection
- Data viewer (GDX)
 - Export
 - Charting
- GAMS Processes Control
- Model Libraries



Free Model Libraries

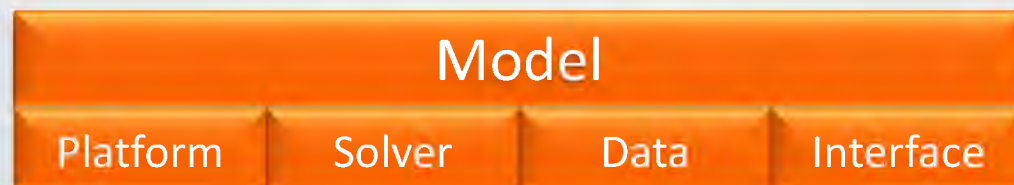


➔ More than 1250 models!



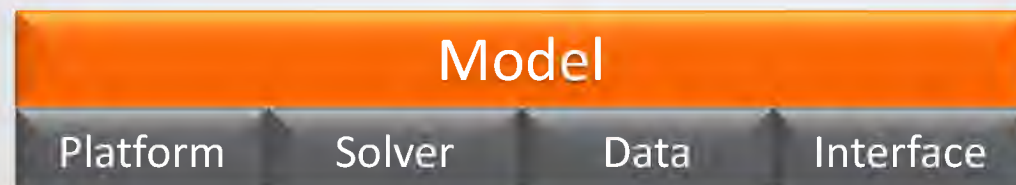
Design Principles

- 1 {
 - Simple modeling language with a balanced mix of declarative and procedural elements
- 2 {
 - Open architecture and interfaces to other systems, independent layers



Simple Declarative Language

- 1 {
 - Few basic language elements: sets, parameters, variables, equations, models
- 2 {
 - Language similar to mathematical notation
- 3 {
 - Easy to learn
- 4 {
 - Model is executable description of the problem
- 5 {
 - Lot's of code optimization under the hood



Mix of Declarative and **Procedural** Elements

Procedural elements like loops, for, if, macros and functions

Allow to build complex problem algorithms within GAMS



Interaction with other systems:

- Job control
- Data exchange

Model

Platform

Solver

Data

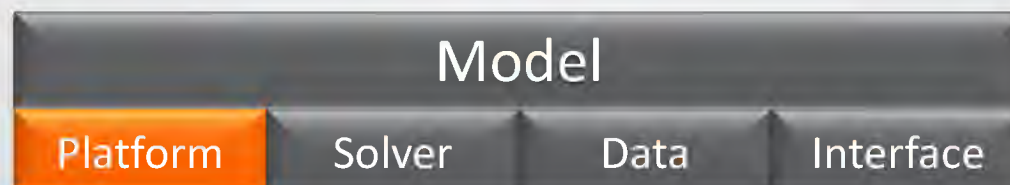
Interface

Independence of **Model** and **Operating System**



Platforms supported by GAMS:

➔ **Models can be moved between platforms with ease!**



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

FICO

Gurobi
Optimization

IBM

mosek

Sulum



➔ Switching between solvers with one line of code!

Model

Platform

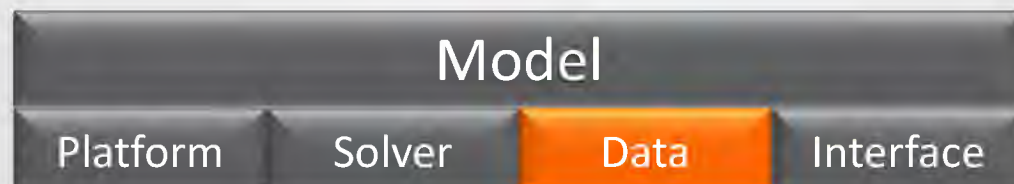
Solver

Data

Interface

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
- No modeling capability: Model is written in GAMS
- Wrapper class that encapsulates a GAMS model



Model

Platform

Solver

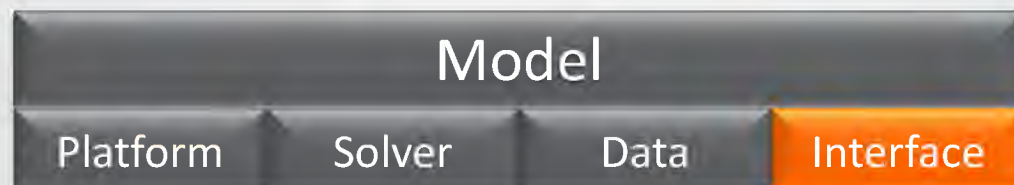
Data

Interface

Simple Encapsulation of a GAMS Model

Very simple interface:

- Properties to **communicate** input data and results
- Properties to **change options** like the solver to use
- Run() method to **run the model**



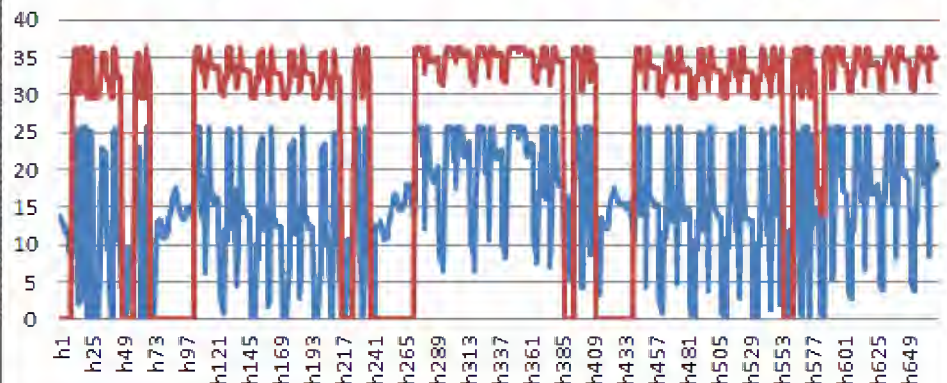
Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
4	Coal	-17.62	-35.24												MWh
5	WasteHeat				36.43	53.2	-42.56	-10.64	-53.2						MWh
6	Steam		5.5	11			10.72			-16.7					kg/s
7	Exhaust									16.7	-16.7				kg/s
8	Heat				18			8.6184			39.84				MWh
9	Electricity	-0.25	-0.5			13.77	29.55			10.39					MWh

Solution - Generated Heat & Electricity



Model

Platform

Solver

Data

Interface

Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

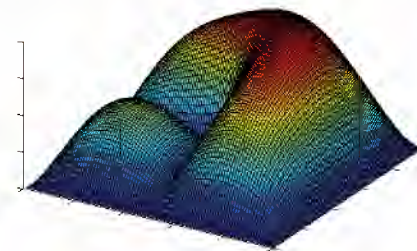
A large orange circle with a dashed border containing the text "MatLab" in white.

Figure 1: US dollar short rate scenarios

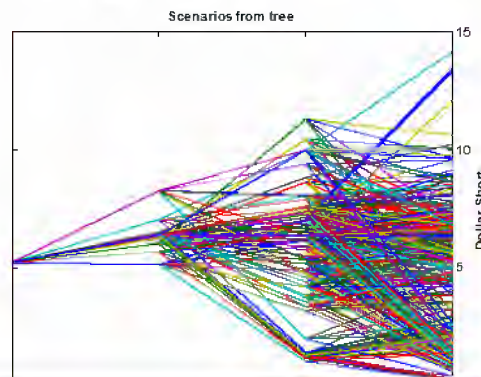
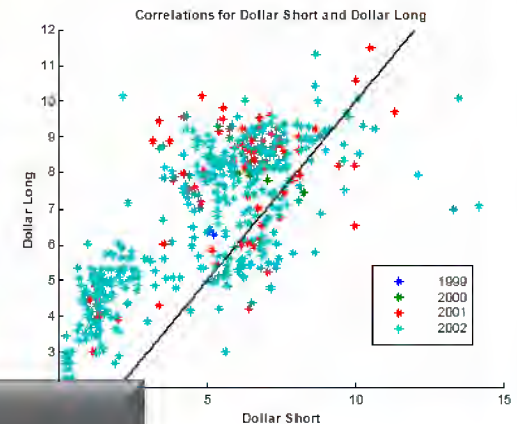


Figure 2: Short vs. long rates



Model

Platform

Solver

Data

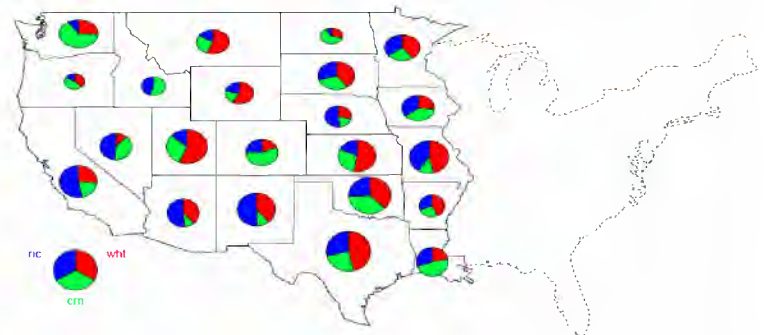
Interface

Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application



```
v <- rgdx(fnSol, list(name = "tour", form = "sparse"))
nxt <- v$val[, 2]
# compute the sequence of cities, based on nxt
solSeq <- NA * c(1:n + 1)
k <- 1
for (j in c(1:n))
  solSeq[j] <-
    k <- nxt[k]
}
solSeq[n + 1] <-
if (k != 1) stop
loc <- cmdscale(e
rx <- range(x <-
ry <- range(y <-
tspres <- loc[sol
s <- seq(n)
```



Model

Platform

Solver

Data

Interface

Striving for **Innovation** and **Compatibility**

Models must benefit from:

Advancing hardware / New Platforms

Enhanced / new solver and solution technology

Improved / upcoming interfaces to other systems

New Modeling Concepts

Protect investments of Users

Life time of a model: 15+ years

New maintainer, platform, solver, user interface

Backward Compatibility

Software Quality Assurance



New Modeling and Solution Concepts

Examples:

- Disjunctive Programs
- Bilevel Programs
- Extended Nonlinear Programs
- Stochastic Programming
- ...

Issues:

- Breakouts of traditional Mathematical Programming classes
- No conventional syntax
- Limited support with common model representation
- Incomplete/experimental solution approaches
- Lack of reliable/any software

➔ **GAMS is conservative when it comes to syntax extensions**

The “GAMS” – Approach

Extended Mathematical Programming

Experimental framework for **automated** mathematical programming **reformulations**

Keep the language simple: Do **not overload** existing GAMS notation

Use **existing language features** to specify additional model features, structure, and semantics

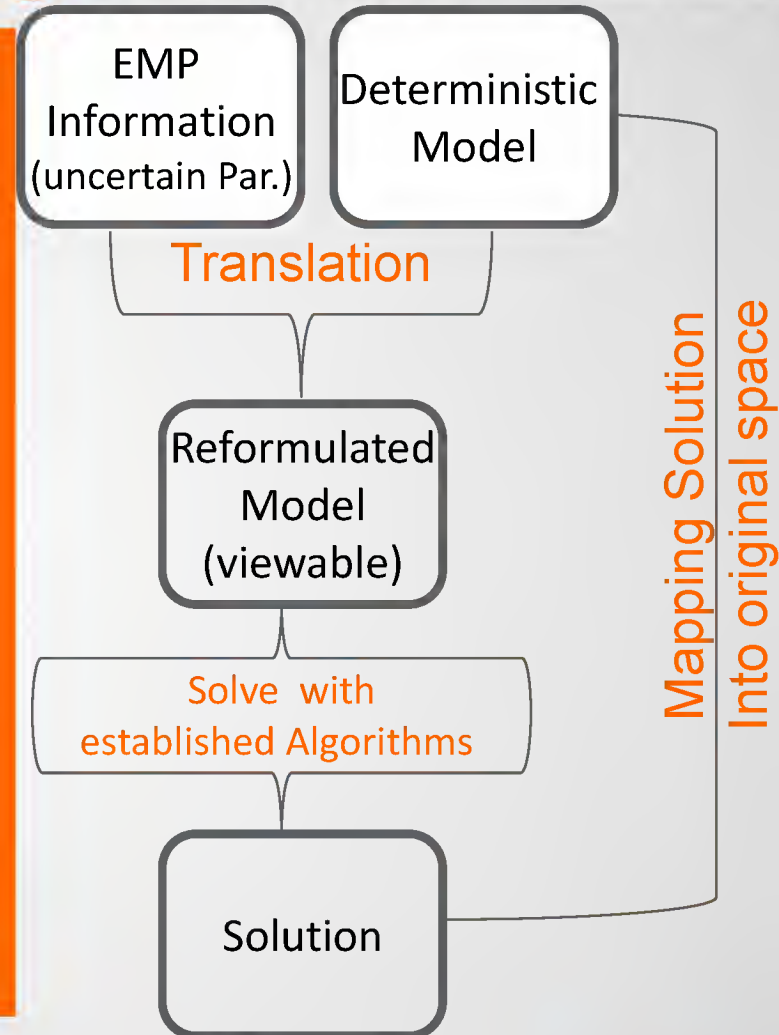
Express **extended model information** in **symbolic (source) form** and apply existing modeling/solution technology

Package new tools with the production system

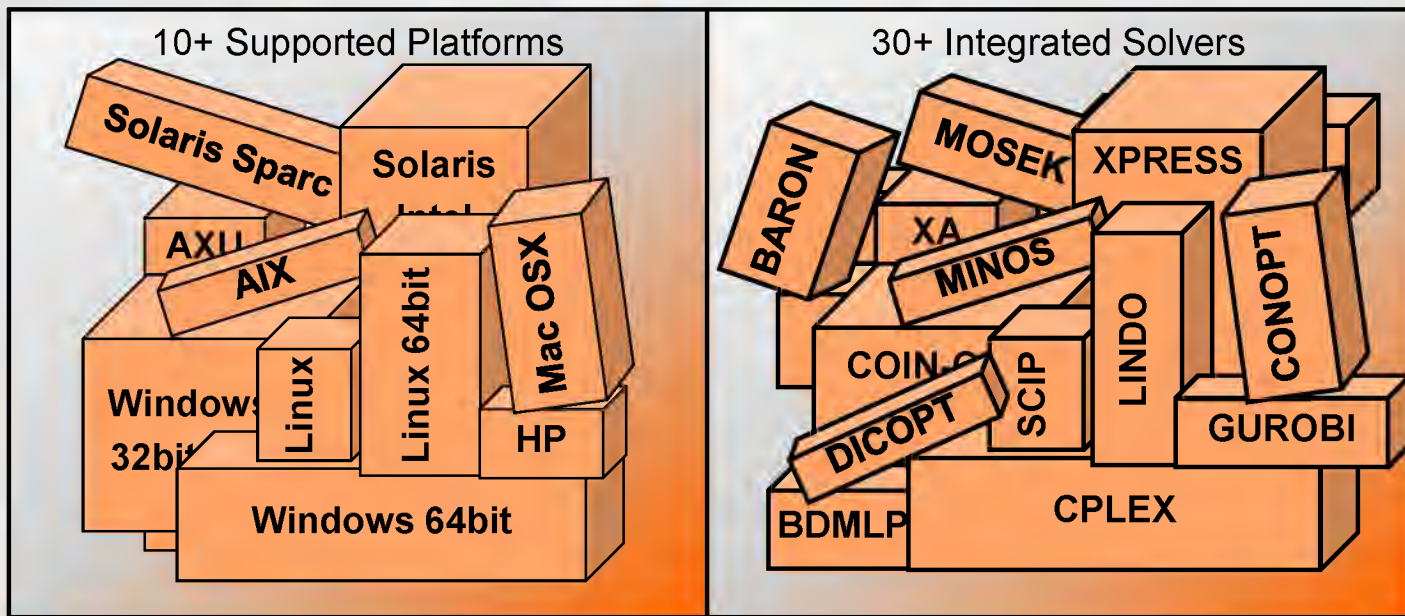
Stochastic Programming in GAMS

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)



Software Quality Assurance



Perspectives:

- What is the impact of new features?
- What is the impact of updated or new solvers?
- Is the new distribution backward compatible?
- ...

Quality Test Models Library

- Tests to verify proper behavior of the system
- More than 600 quality test models, each containing numerous pass/fail tests
- Automatically executed every night for all solver combinations: → 12,000 runs / platform (all tests)
- Automatically generated test summaries with different level of information
- Assurance about the basic functionality of the software!

Latest GAMS System Builds and Test Results

Sunday 23Mar14 04:52 (UTC)

[[Latest Builds](#) | [Alpha Builds](#) | [Beta Builds](#) | [Nightly Builds](#) | [Glossary](#)]

[Comments?](#)

NOTE: The (nightly) alpha builds are internal development versions of the GAMS system. They may have known bugs, unfinished features, beta versions of third-party software, or may not function at all! Not for production use!

nightly α	System	Libraries	Build	Rev	Status and Time (UTC)	Initial Tests	Full Tests
Friday	lnx	Download	24.3.0	45152	Test done 22Mar2014 12:42:54	805 runs 0 failures (q=0,s=0)	Report 11780 runs 0 failures (q=0,s=0) Report
Saturday	leg	Download	24.3.0	45152	Test done 22Mar2014 22:03:49	804 runs 0 failures (q=0,s=0)	Report 2198 runs 0 failures (q=0,s=0) Report
Friday	vs8	Download	24.3.0	45152	Test started 22Mar2014 01:12:52	947 runs 0 failures (q=0,s=0)	Report results pending
Friday	wei	Download	24.3.0	45152	Test started 22Mar2014 03:47:14	944 runs 1 failures (q=1,s=0)	Report results pending

Why **GAMS**?

- Experience of 25+ 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

Agenda

What is GAMS?

A Simple Example

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 how easy it is to change model type and, consequently, solver technology

Model types **in this example**

LP

- Determine minimum transportation cost.
Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
e.g. if we ship, then we ship at least 100 cases.

MINLP

- Allows non-linearity,
e.g. a smooth decrease in unit cost when
shipping volumes grows

SP

- Allows uncertainty,
e.g. uncertain demand

A Simple **Transportation Problem**

Canning Plants (supply)

shipments

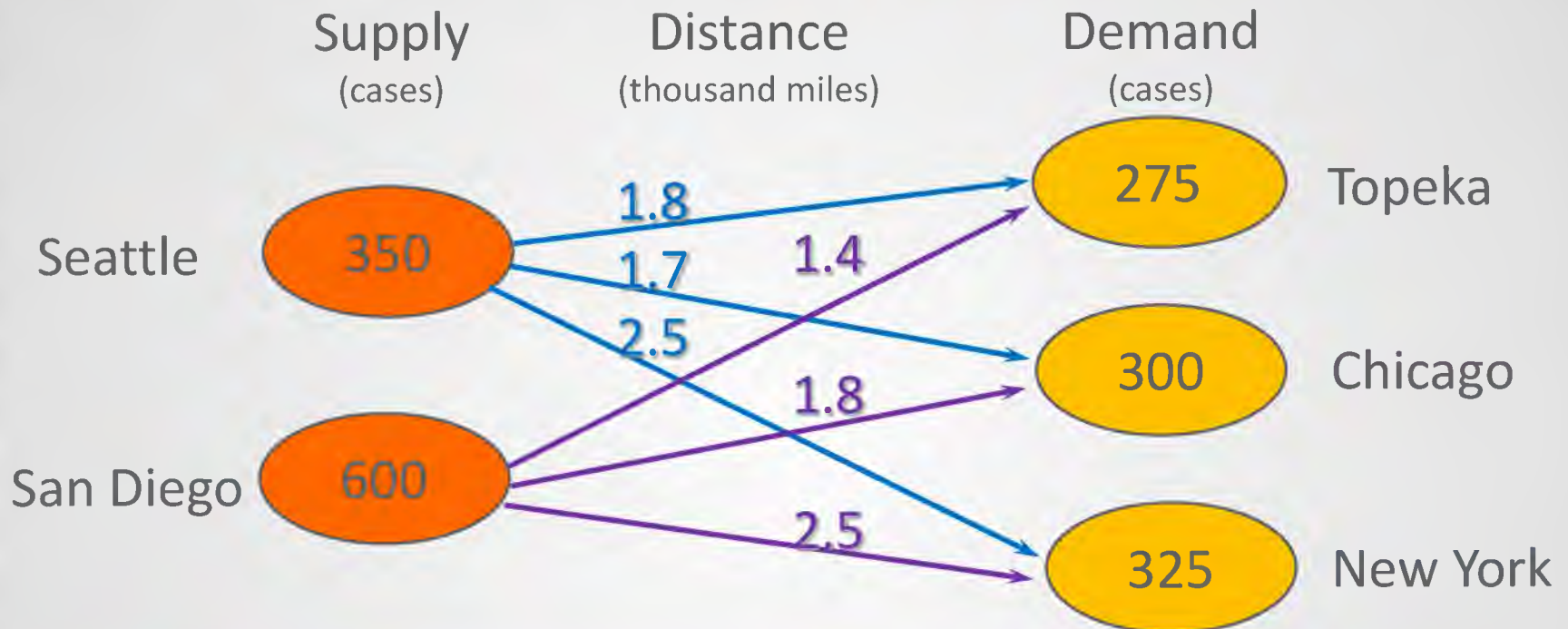
(Number of cases)

Markets (demand)



Freight: \$90 case / thousand miles

A Transportation Model



Minimize
subject to

Transportation cost
Demand satisfaction at markets
Supply constraints

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_j (Demand in cases)

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

subject to

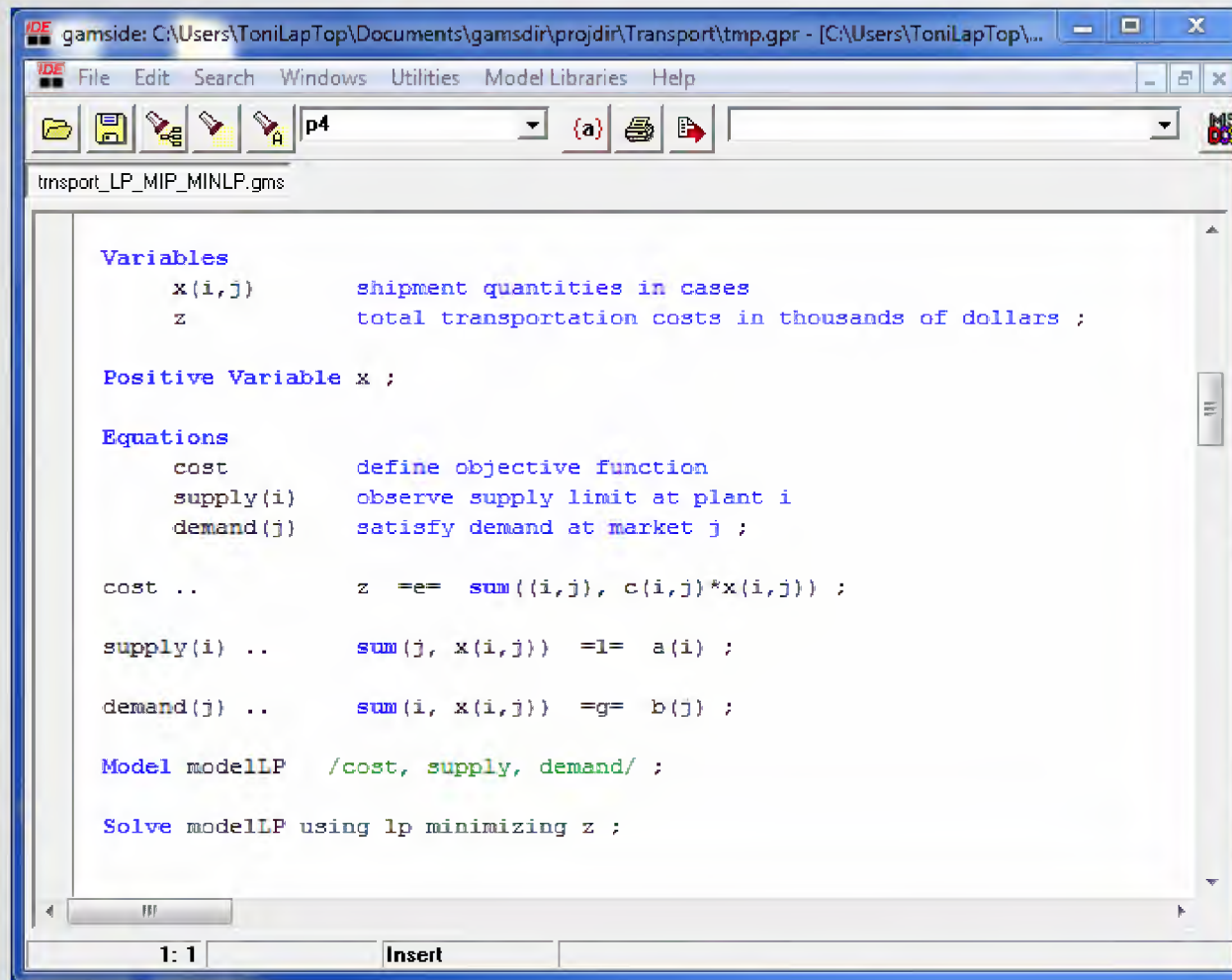
$\sum_j x_{ij} \leq a_i \quad \forall i$ (Shipments from each plant $\leq$ supply capacity)

$\sum_i x_{ij} \geq b_j \quad \forall j$ (Shipments to each market $\geq$ demand)

$x_{ij} \geq 0 \quad \forall i, j$ (Do not ship from market to plant)

$i, j \in \mathbb{N}$

GAMS Algebra (LP Model)



The screenshot shows the GAMS IDE window titled "gamside: C:\Users\ToniLapTop\Documents\gamsdir\projdir\Transport\tmp.gpr - [C:\Users\ToniLapTop\...". The menu bar includes File, Edit, Search, Windows, Utilities, Model Libraries, and Help. The toolbar contains icons for file operations and a dropdown menu showing "p4". The main text area displays the following GAMS code:

```
transport_LP_MIP_MINLP.gms

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 modellP /cost, supply, demand/ ;

Solve modellP using lp minimizing z ;
```

The status bar at the bottom shows "1: 1" and "Insert".

Hands-On

GAMS Algebra (Data Input)

The screenshot shows the GAMS IDE interface. The main window displays the 'data.gms' file with the following content:

```

Sets
  i  canning plants   / seattle, san-diego /
  j  markets          / new-york, chicago, topeka / ;

Parameters
  a(i)  capacity of plant i in cases
        / seattle    350
          san-diego  600 /

  b(j)  demand at market j in cases
        / new-york   325
          chicago    300
          topeka     275 / ;

Table d(i,j)  distance in thousands of miles
           new-york    chicago    topeka
seattle    2.5         1.7         1.8
san-diego  2.5         1.8         1.4 ;

Scalar f  freight in dollars per case per thousand miles /90/ ;

Parameter c(i,j)  transport cost in thousands of dollars per case ;

               c(i,j) = f * d(i,j) / 1000 ;

```

The bottom window shows the model structure:

```

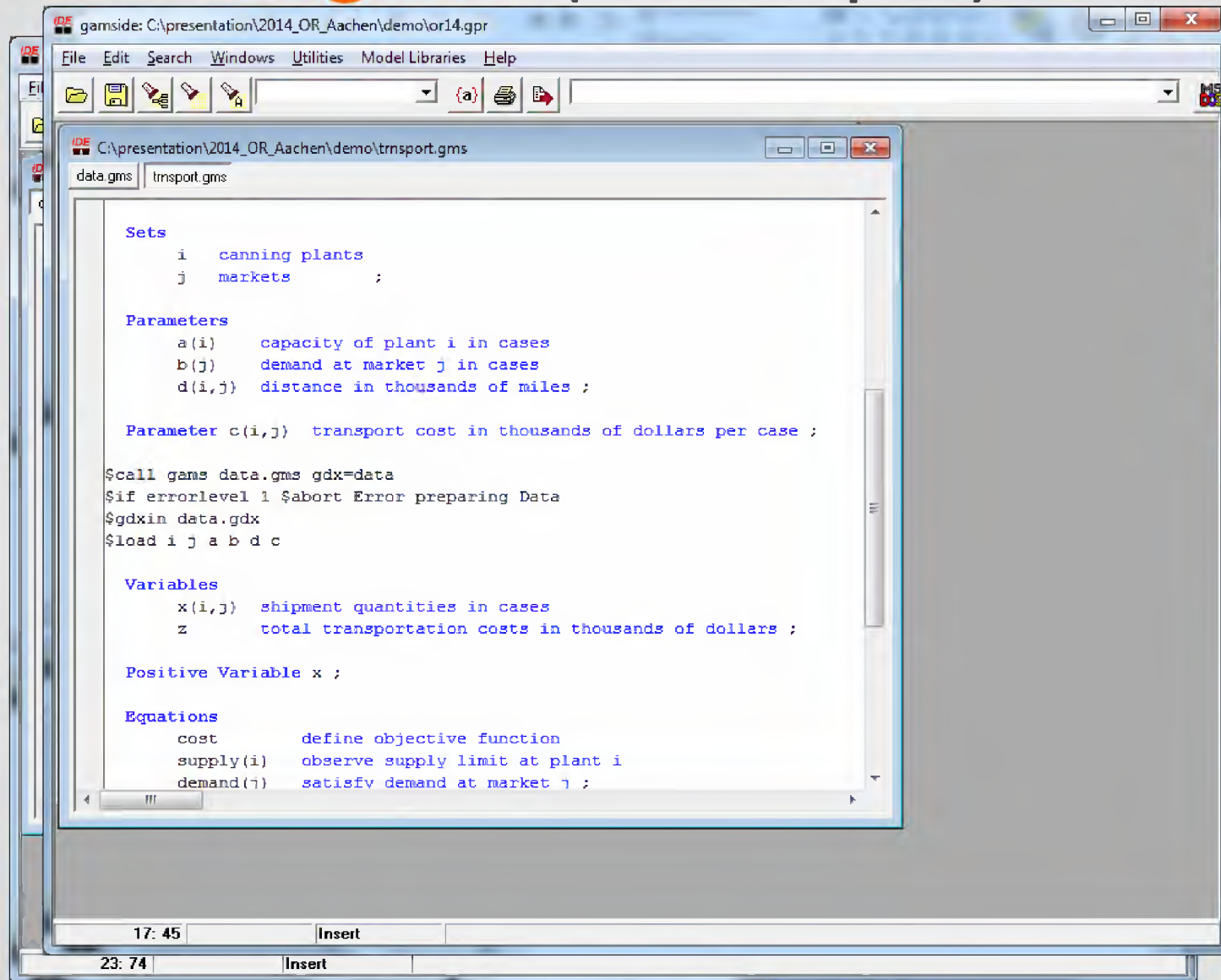
$include data.gms

Variables
  x(i,j)  shipment quantities in cases

```

The status bar at the bottom indicates '23: 74' and 'Insert'.

GAMS Algebra (Data Input)



```
IDE gamside: C:\presentation\2014_OR_Aachen\demo\or14.gpr
File Edit Search Windows Utilities Model Libraries Help
C:\presentation\2014_OR_Aachen\demo\transport.gms
data.gms transport.gms

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 ;

Parameter c(i,j)  transport cost in thousands of dollars per case ;

$call gams data.gms gdx=data
$if errorlevel 1 $abort Error preparing Data
$gdxin data.gdx
$load i j a b d c

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 ;
```

17: 45 Insert

23: 74 Insert

GAMS Algebra (Data Input)

The screenshot displays the GAMS IDE interface with a file named 'transport.gms' open. The code in the file defines sets, parameters, and variables for a transportation problem. An Excel spreadsheet 'data.xlsx' is also open, showing data for the same problem.

GAMS Model Code (transport.gms):

```

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 ;

$call gdxrw data.xlsx par=d rng=A1 par=a rng=G2 dim=1 rdim=1 par=b rng=B6 dim=1
$if errorlevel 1 $abort Error preparing Data
$gdxin data.gdx
$load i<d.dim1 j<d.dim2 d a b

Scalar f  freight in dollars per case per thousand
Parameter c(i,j)  transport cost in thousands of
    c(i,j) = f * d(i,j) / 1000 ;

Variables
    x(i,j)  shipment quantities in cases
    z       total transportation costs in thousand dollars

Positive Variable x ;

Equations
    cost      define objective function
    supply(i) observe supply limit at plant i
  
```

Excel Spreadsheet (data.xlsx):

	A	B	C	D	E	F	G	H
1		New-York	Chicago	Topeka				
2	Seattle	2.5	1.7	1.8		Seattle	350	
3	San-Diego	2.5	1.8	1.4		San-Diego	600	
4								
5								
6		New-York	Chicago	Topeka				
7		325	300	275				
8								

Solution to LP model

Canning Plants (supply)

shipments

(Number of cases)

Markets (demand)



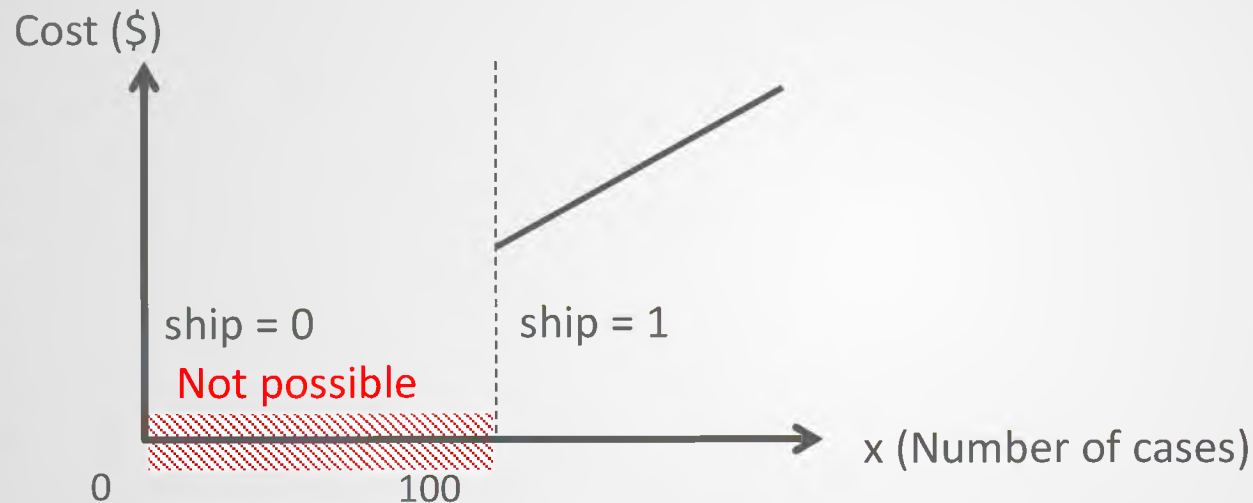
Freight: \$90 case / thousand miles

Total cost: \$153,675



Minimum **shipment of 100 cases**

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



add constraints:

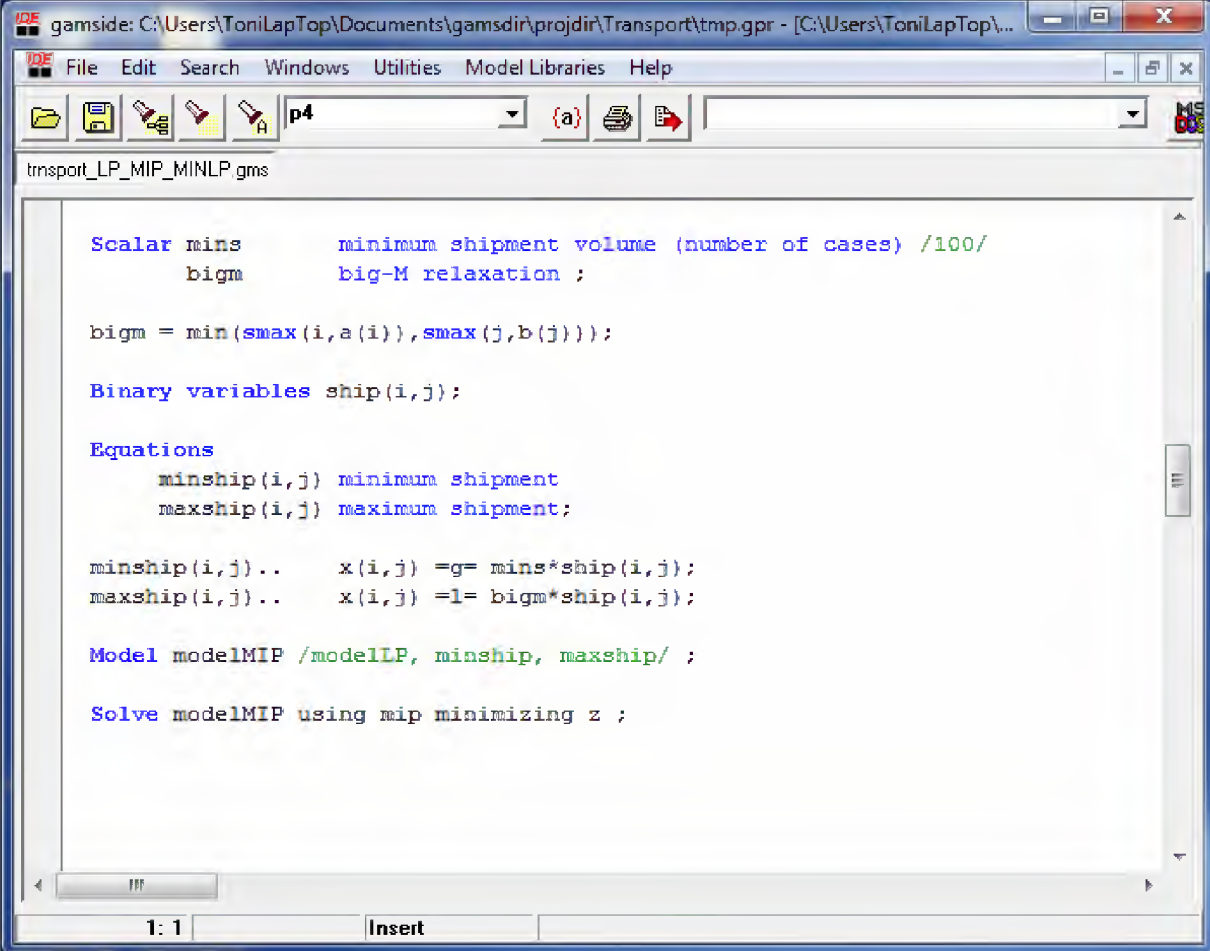
$$x_{i,j} \geq 100 \cdot \text{ship}_{i,j} \quad \forall i,j \quad (\text{if } \text{ship}=1, \text{ then ship at least 100})$$

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

$$\text{ship}_{i,j} \in \{0,1\}$$

Hands-On

GAMS Algebra (MIP Model)



The screenshot shows the GAMS IDE window titled "gamside: C:\Users\ToniLapTop\Documents\gamsdir\projdir\Transport\tmp.gpr - [C:\Users\ToniLapTop\...". The menu bar includes File, Edit, Search, Windows, Utilities, Model Libraries, and Help. The toolbar contains icons for file operations and a dropdown menu showing "p4". The main text area displays the following GAMS code:

```
transport_LP_MIP_MINLP.gms

Scalar mins          minimum shipment volume (number of cases) /100/
      bigm            big-M relaxation ;

bigm = min(smax(i,a(i)),smax(j,b(j)));

Binary variables ship(i,j);

Equations
    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 modelMIP /modelLP, minship, maxship/ ;

Solve modelMIP using mip minimizing z ;
```

The status bar at the bottom shows "1: 1" and "Insert".

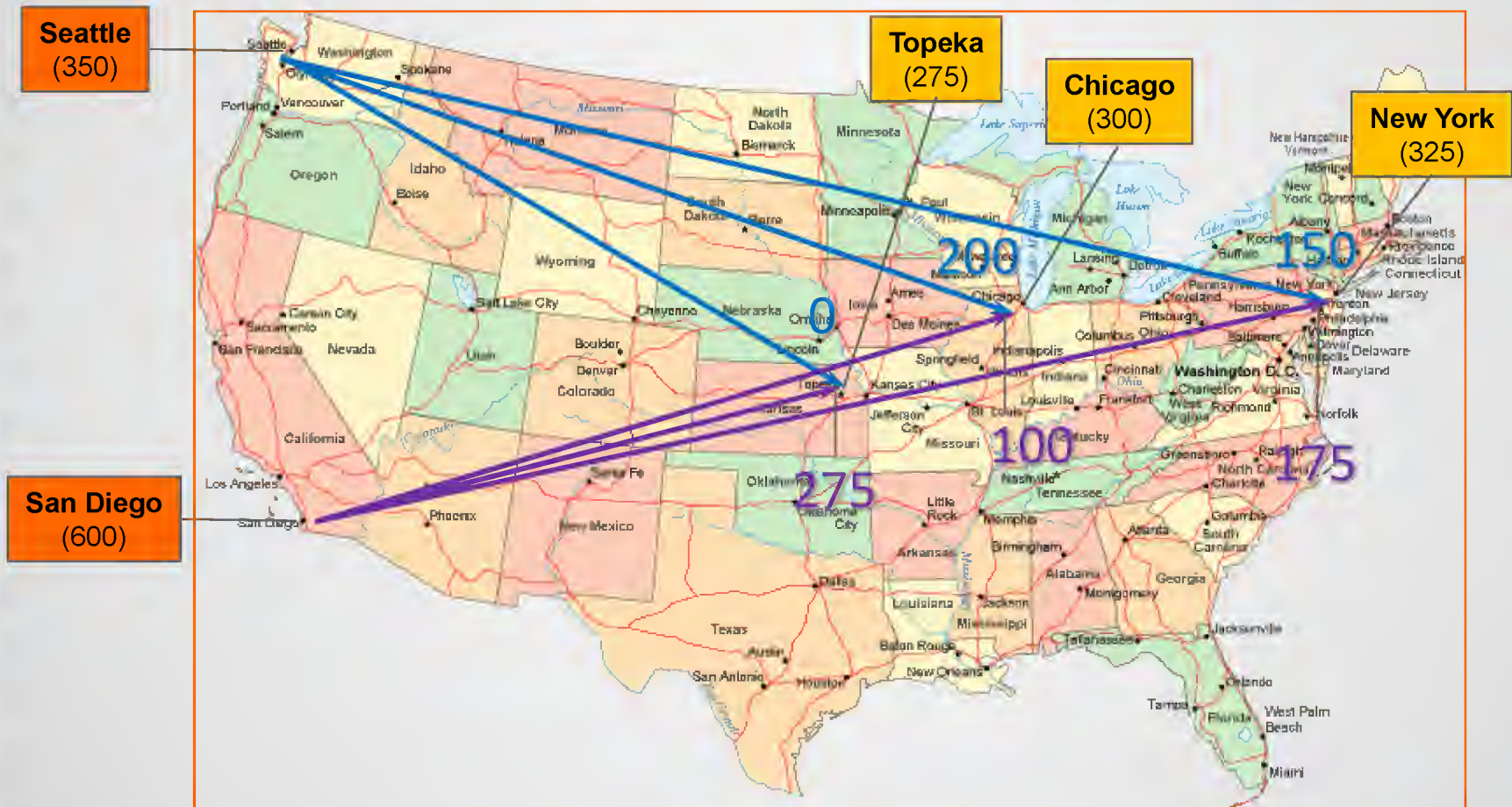
Solution to **MIP model**

Canning Plants (supply)

shipments

(Number of cases)

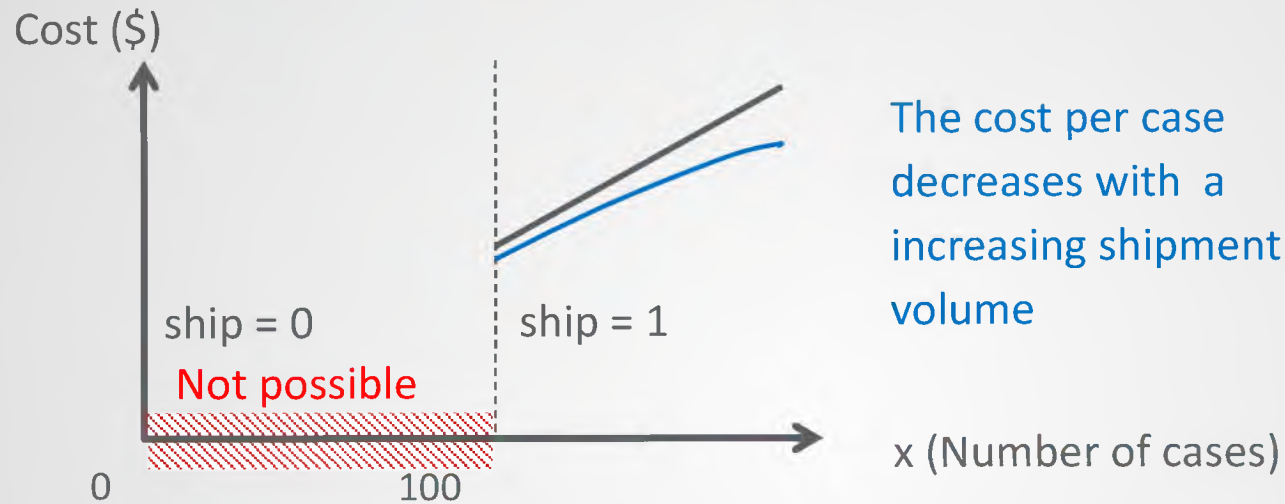
Markets (demand)



Freight: \$90 case / thousand miles

Total cost: \$153,675

Cost Savings (non-linear)



Replace:

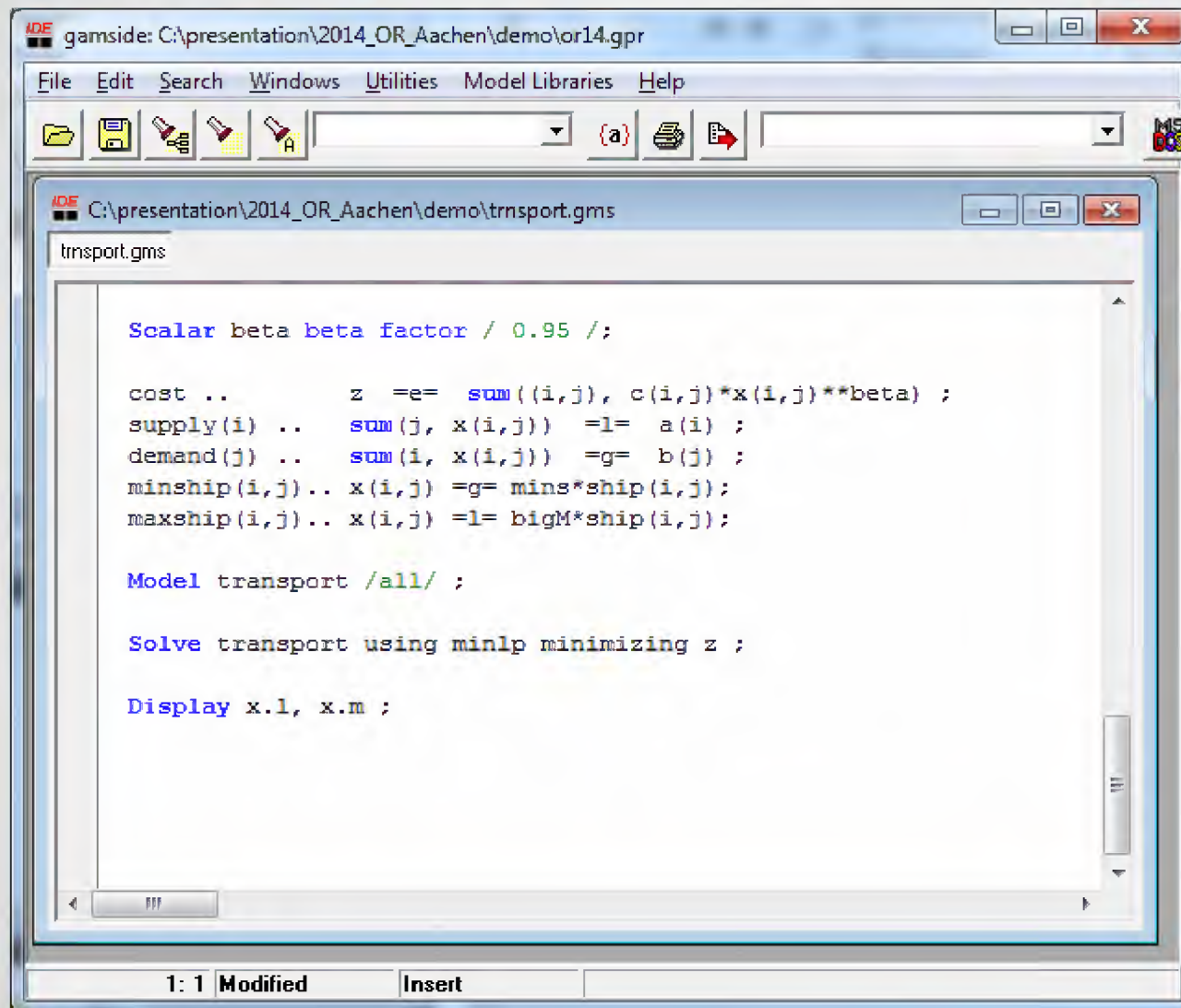
min $\sum_i \sum_j c_{ij} \cdot x_{ij}$ (Minimize total transportation cost)

With

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

Hands-On

GAMS Algebra (MINLP Model)



The screenshot shows the GAMS IDE interface. The main window displays the code for a model named 'transport.gms'. The code defines a scalar 'beta' with a value of 0.95, a cost function 'z' as the sum of costs multiplied by flow and beta, supply and demand constraints, minimum and maximum shipment constraints, and solves the model using the MINLP solver.

```
Scalar beta beta factor / 0.95 /;

cost ..      z =e= sum((i,j), c(i,j)*x(i,j)**beta) ;
supply(i) .. sum(j, x(i,j)) =l= a(i) ;
demand(j) .. sum(i, x(i,j)) =g= b(j) ;
minship(i,j).. x(i,j) =g= mins*ship(i,j);
maxship(i,j).. x(i,j) =l= bigM*ship(i,j);

Model transport /all/ ;

Solve transport using minlp minimizing z ;

Display x.l, x.m ;
```

Solution to **MINLP model**

Canning Plants (supply)

shipments

(Number of cases)

Markets (demand)

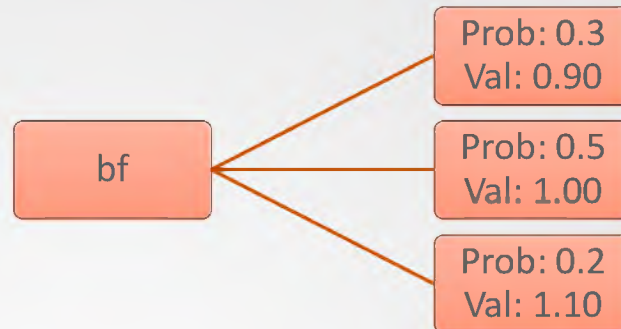


Freight: ~\$90 case / thousand miles

Total cost: \$115,438

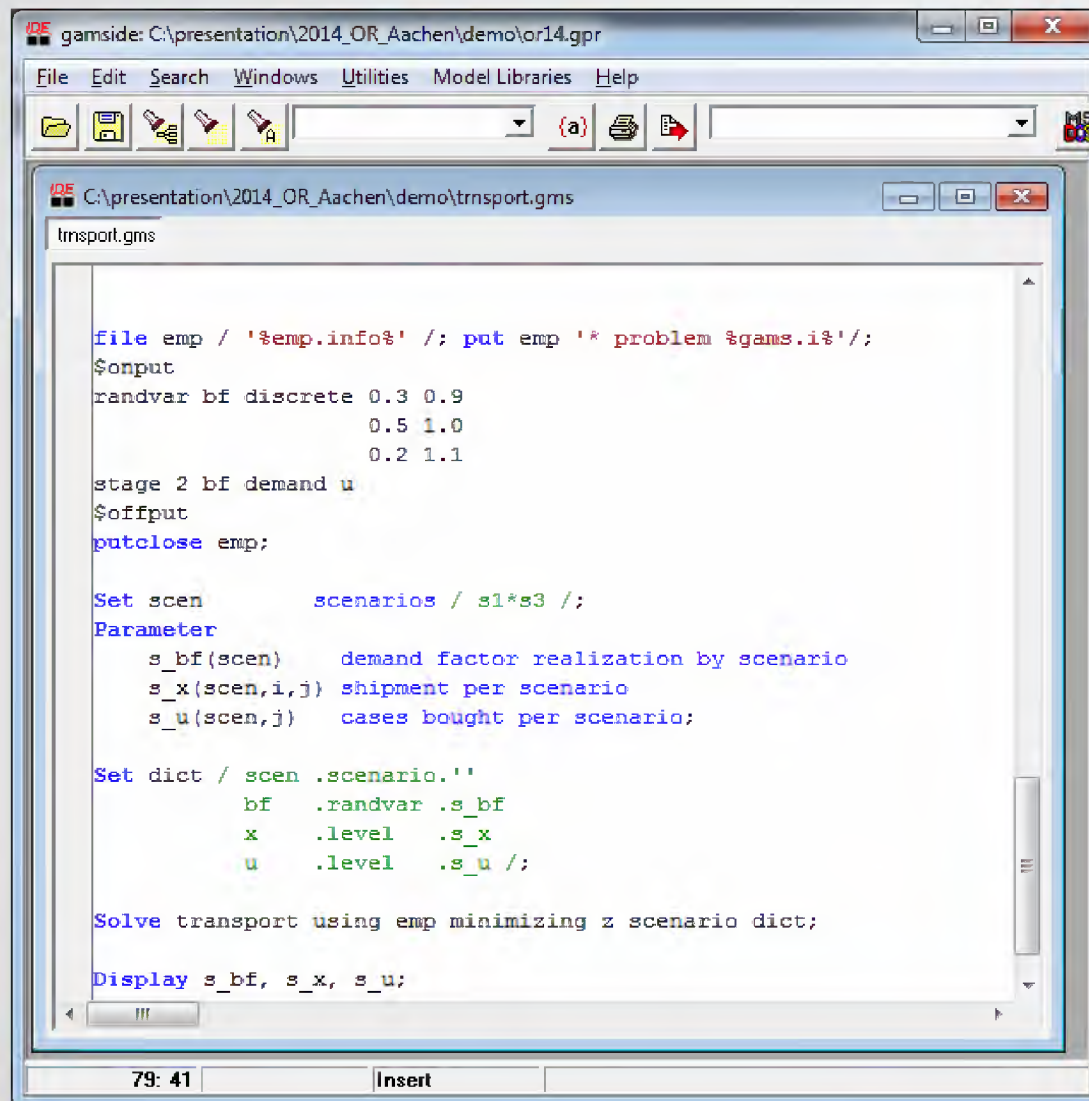
Uncertain Demand

- Uncertain demand factor *bf*



- Decisions to make:
 - How many units should be shipped “here and now” (without knowing the outcome of the uncertain demand)?
→ *First-stage decision*
 - What can be done after the outcome becomes known and we did not ship enough?
→ *Second-stage or recourse decision*
 - Recourse decisions can be seen as
 - penalties for bad first-stage decisions
 - variables to keep the problem feasible

GAMS Algebra (SP Model)



The screenshot shows the GAMS IDE interface. The title bar indicates the file path: C:\presentation\2014_OR_Aachen\demo\or14.gpr. The menu bar includes File, Edit, Search, Windows, Utilities, Model Libraries, and Help. The toolbar contains icons for file operations and execution. The main editor window displays the file C:\presentation\2014_OR_Aachen\demo\transport.gms. The code in the editor is as follows:

```
transport.gms

file emp / '%emp.info%' /; put emp '* problem %gams.i%';
$onput
randvar bf discrete 0.3 0.9
                0.5 1.0
                0.2 1.1

stage 2 bf demand u
$offput
putclose emp;

Set scen          scenarios / s1*s3 /;
Parameter
    s_bf(scen)     demand factor realization by scenario
    s_x(scen,i,j)  shipment per scenario
    s_u(scen,j)    cases bought per scenario;

Set dict / scen .scenario.'
           bf .randvar .s_bf
           x .level .s_x
           u .level .s_u /;

Solve transport using emp minimizing z scenario dict;

Display s_bf, s_x, s_u;
```

The status bar at the bottom shows the line number 79: 41 and the Insert mode.

Solution to **SP model**

Canning Plants (supply)

shipments

(Number of cases)

Markets (demand)



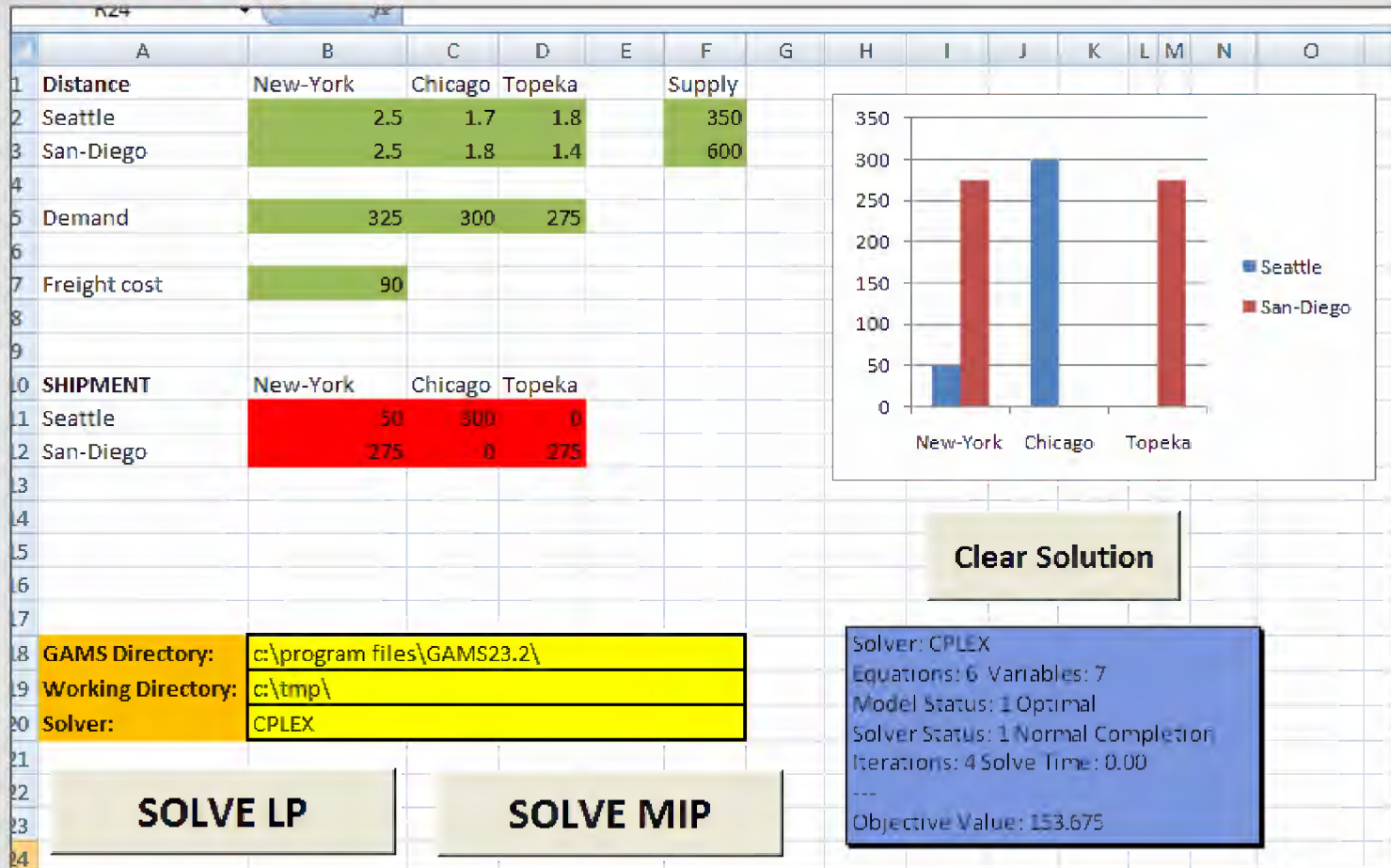
Freight: \$90 case / thousand miles

Total cost: \$158,588

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:
<http://www.gams.com/dd/docs/solvers/empsp.pdf>

Outlook: Running GAMS in an Application


Hands-On



GAMS

Thank You

Europe

GAMS Software GmbH
P.O. Box 40 59
50216 Frechen, Germany
Phone: +49 221 949 9170
Fax: +49 221 949 9171
info@gams.de

USA

GAMS Development Corp. 1217 Potomac
Street, NW Washington, DC 20007
USA
Phone: +1 202 342 0180
Fax: +1 202 342 0181
sales@gams.com