GAMS

Design Principles that Make the Difference

Toni Lastusilta: tlastusilta@gams.com

Franz Nelissen: fnelissen@gams.com

GAMS

Company Background

Roots: World Bank, 1976

Went commercial in 1987

GAMS Development Corporation (Washington)

GAMS Software GmbH (Cologne, Braunschweig) 1996

Tool Provider: **G**eneral **A**lgebraic **M**odeling **S**ystem

GAMS

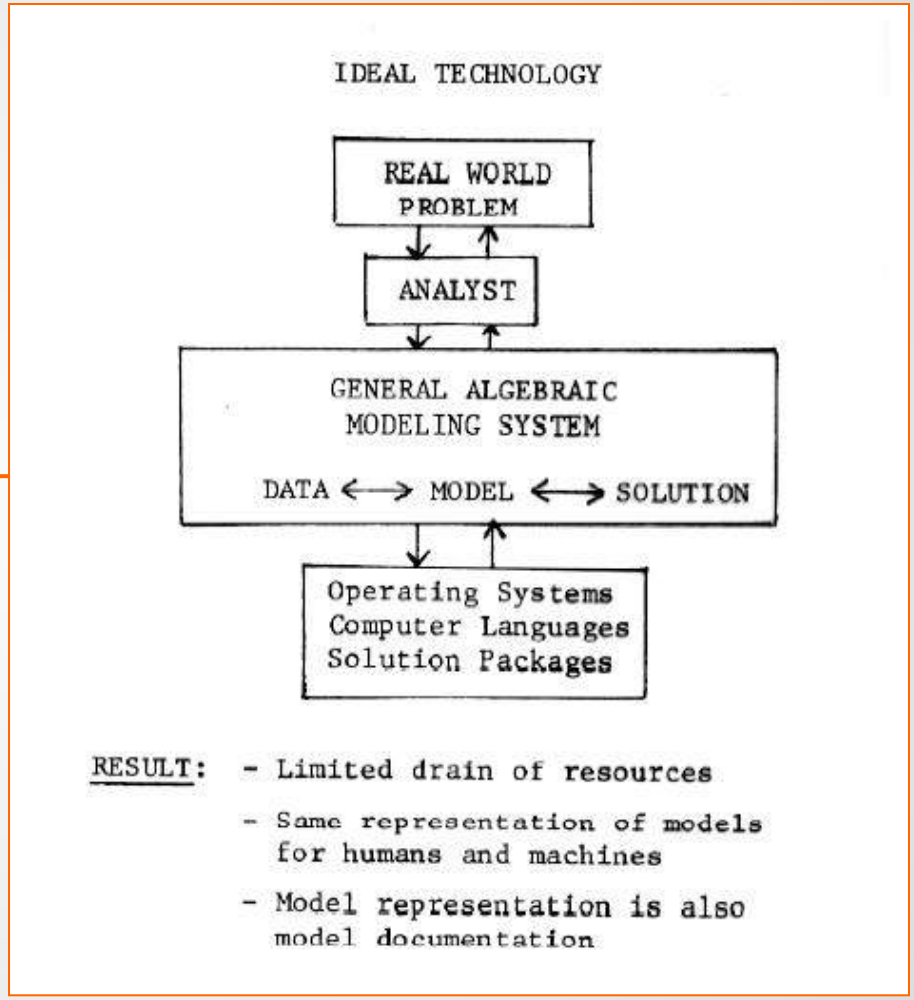
Agenda

- 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
- Some Applications

GAMS

1976 - A World Bank Slide

The Vision



GAMS

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

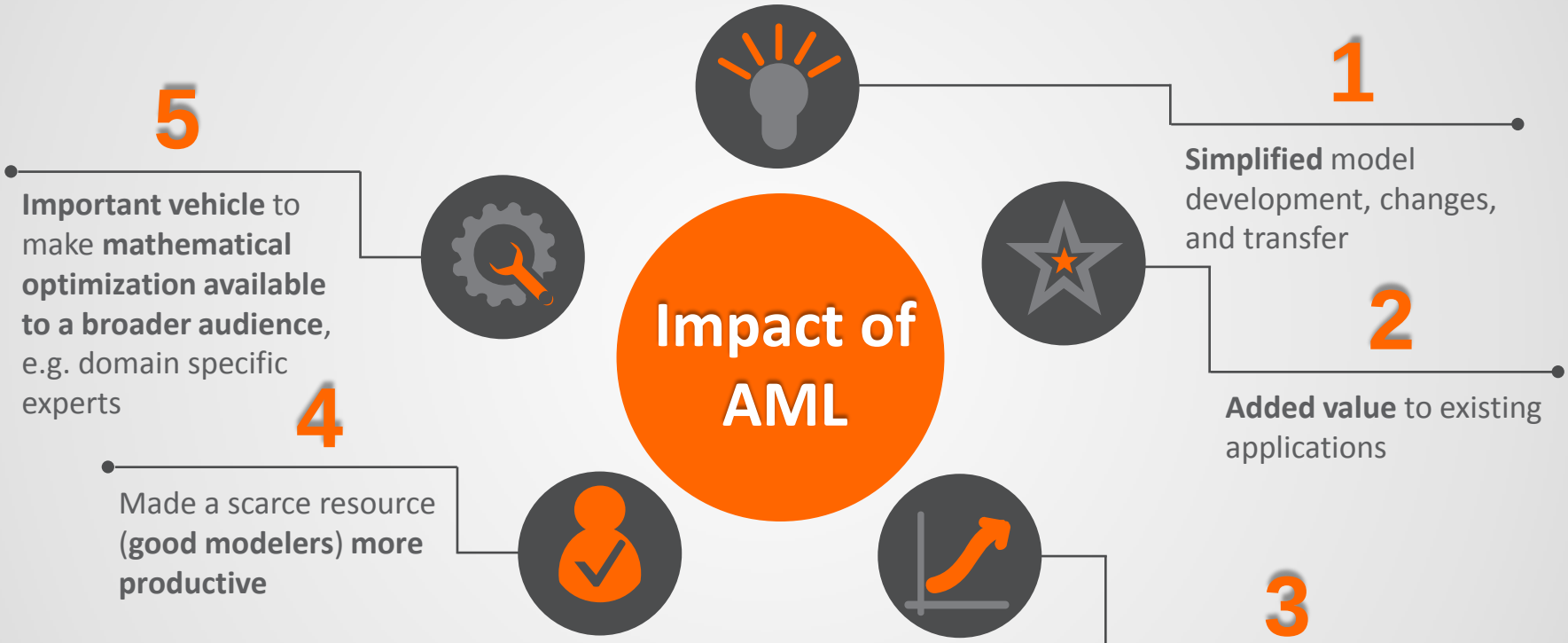
2012 INFORMS Impact Prize

36 Years later

Originators of Algebraic Modeling Languages

GAMS

Impact of Algebraic Modeling Languages



GAMS

What does he have to think about?

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

Why should he use GAMS?

GAMS

Broad User Community and Network

GAMS used in more than 120 countries

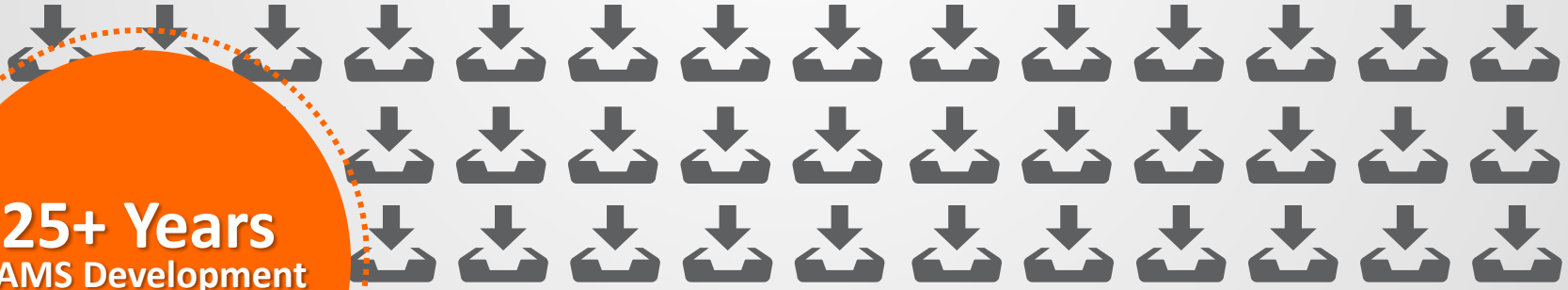
25+ Years
GAMS Development

GAMS

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

GAMS

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

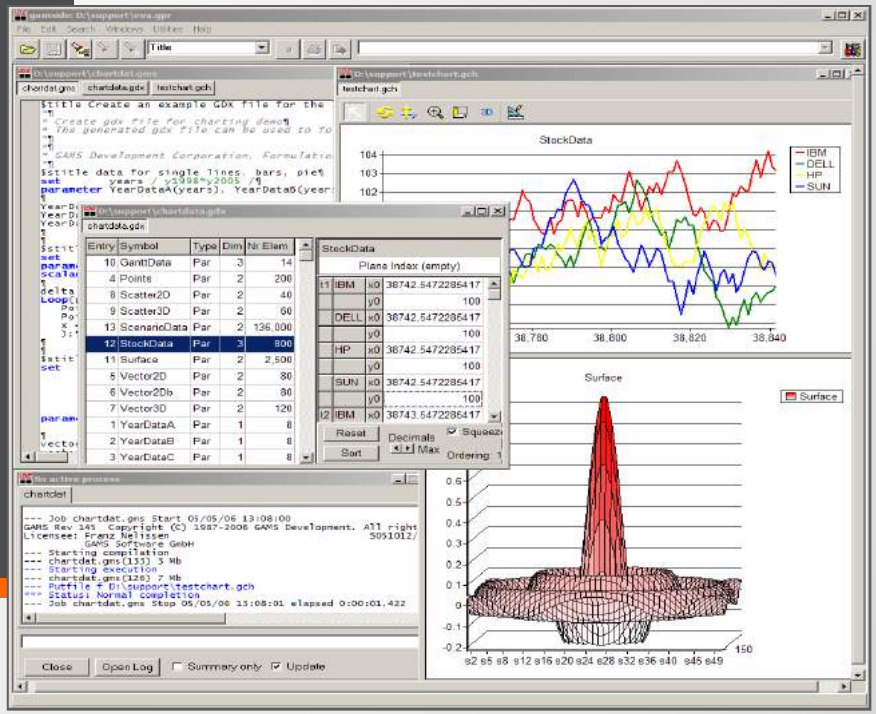
25+ Years
GAMS Development

GAMS

Strong Development Environment

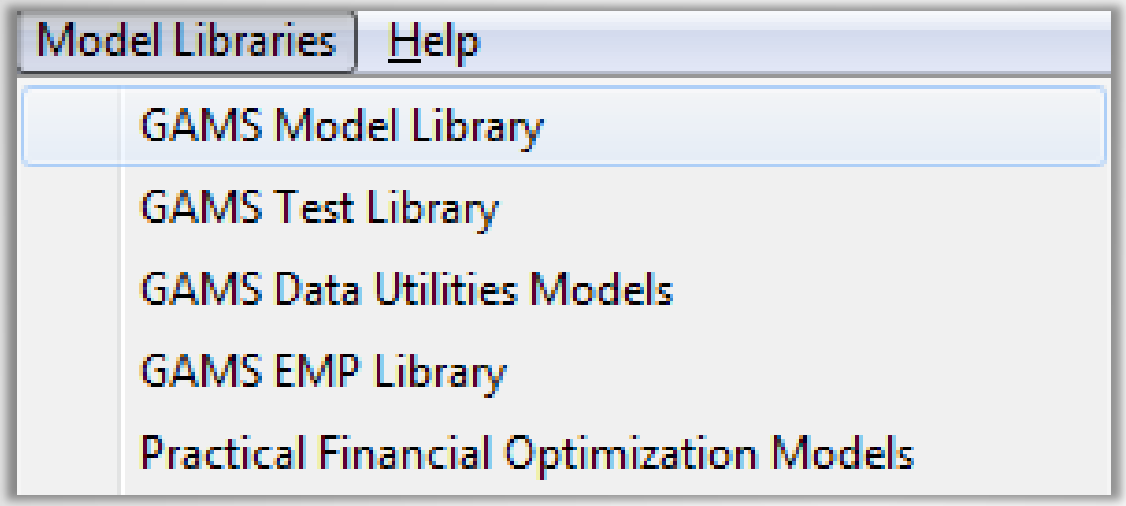
GAMS IDE

- 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

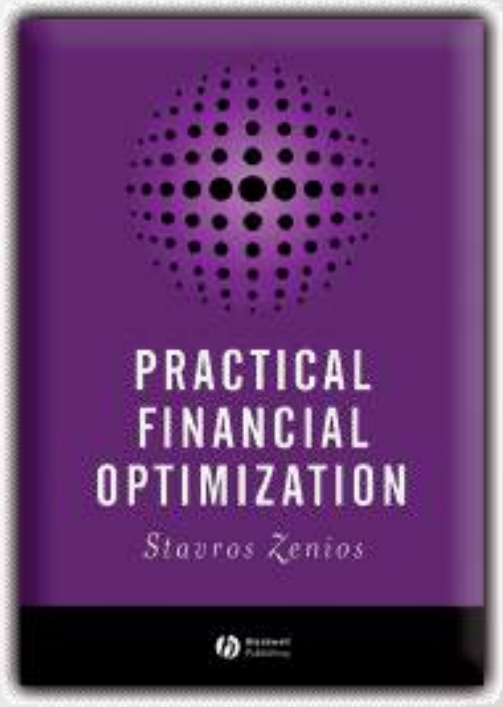


GAMS

Free Model Libraries



➡ More than 1250 models!



GAMS

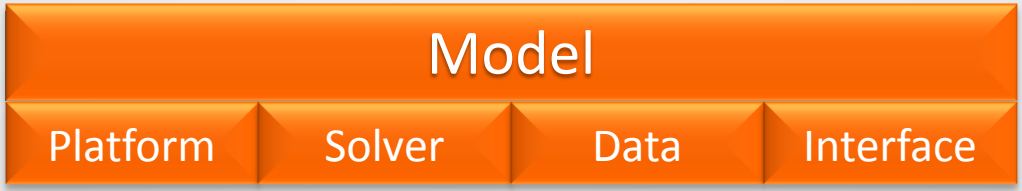
Design Principles

1

- **Simple modeling language with a balanced mix of declarative and procedural elements**

2

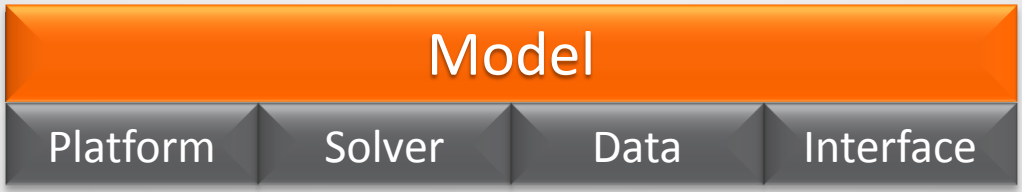
- **Open architecture and interfaces to other systems, independent layers**



GAMS

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



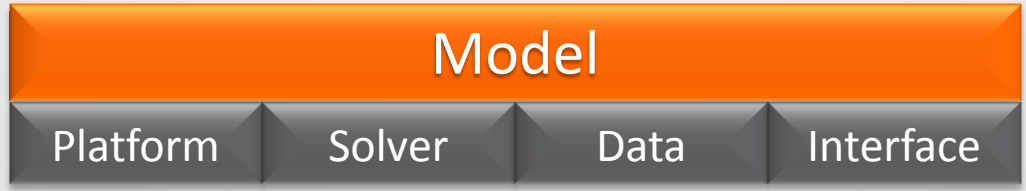
Mix of Declarative and Procedural Elements

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

Allow to build complex problem algorithms within GAMS

Interaction with other systems:

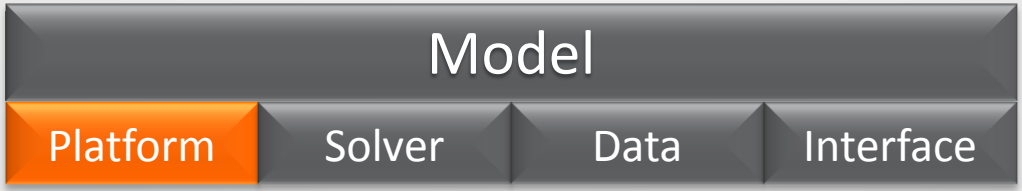
- Job control
- Data exchange



Independence of Model and Operating System

Platforms supported by GAMS:

➡Models can be moved between platforms with ease!



GAMS

Independence of Model and Solver

One environment for a wide range of model types and solvers

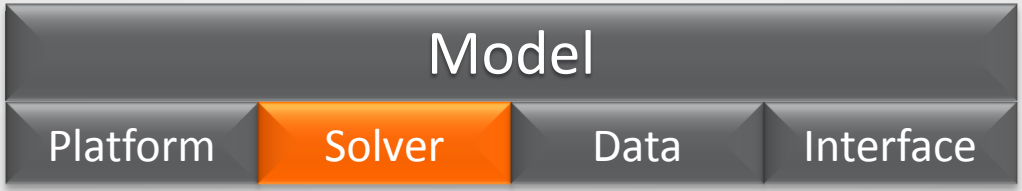
All major commercial LP/MIP solver

Open Source Solver (COIN)

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

FICO · Gurobi Optimization · IBM · mosek · Sulum Optimization · COIN-OR

Switching between solvers with one line of code!

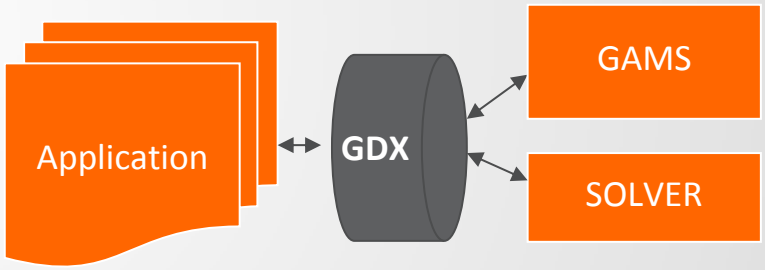


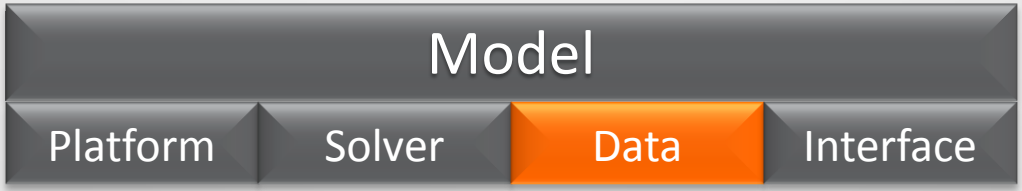
GAMS

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
  - …

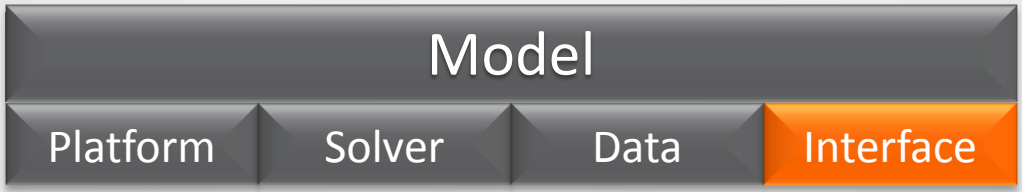




Independence of Model and User Interface

API's

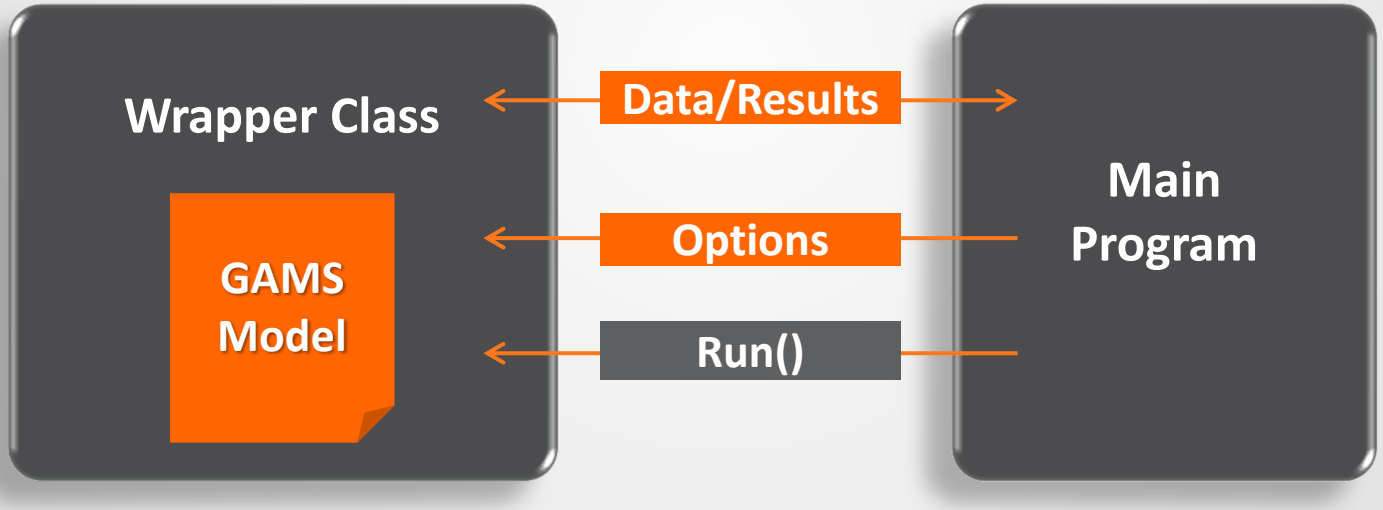
- *Low Level*
- **Object Oriented**: .Net, Java, Python
- No modeling capability: Model is written in GAMS
- Wrapper class that encapsulates a GAMS model

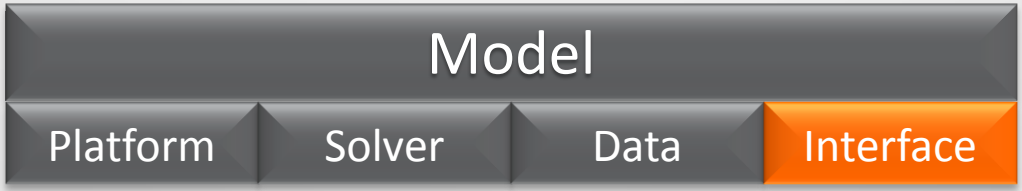


Simple Encapsulation of a GAMS Model

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**





GAMS

Smart Links to other Applications

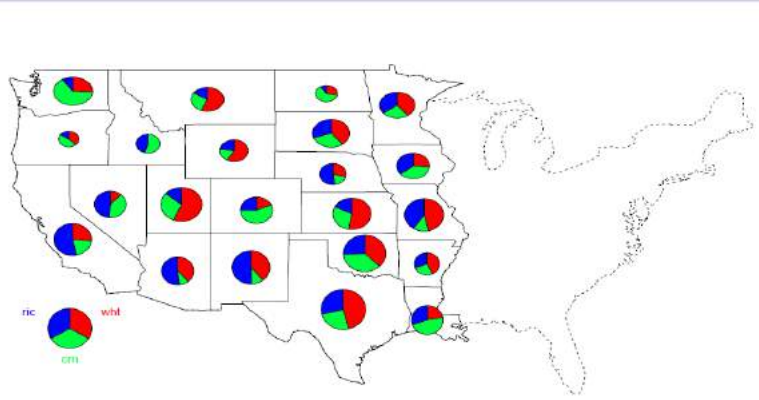
- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

MS Excel

MATLAB

R

```
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(
rx <- range(x <-
ry <- range(y <-
tspres <- loc[so
s <- seq(n)
```



Model

Platform | Solver | Data | Interface

GAMS

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**

GAMS

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

GAMS

The "GAMS" – Approach

Extended Mathematical Programming

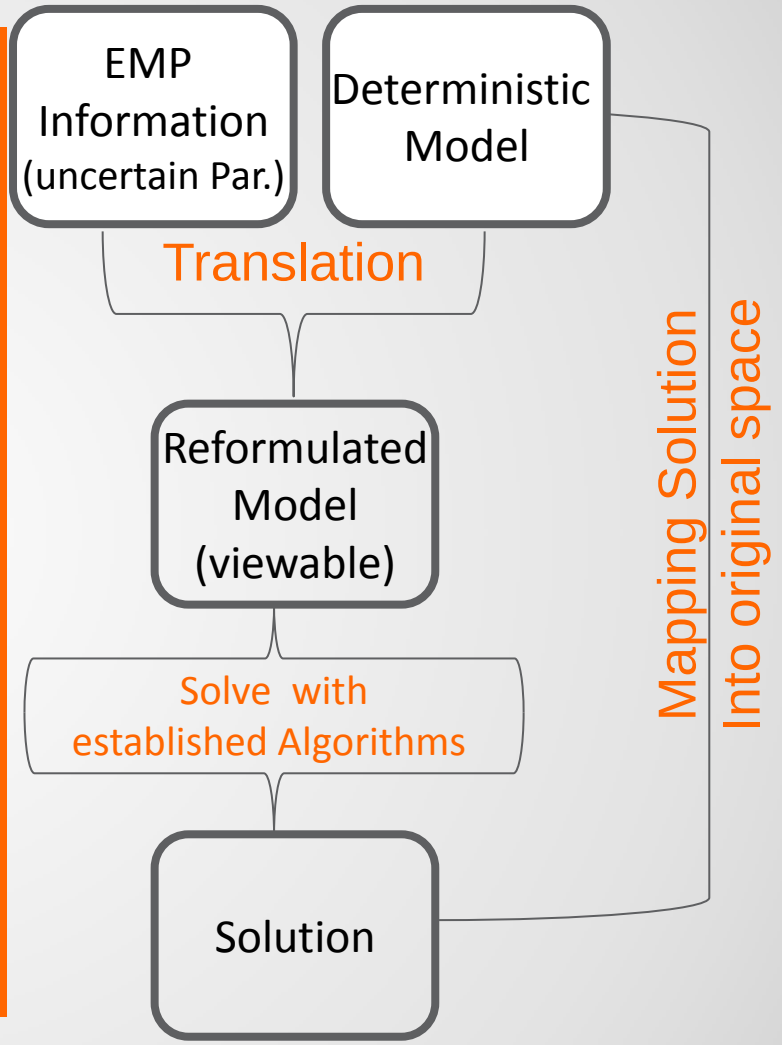
- **Experimental framework** for **automated** mathematical programming **reformulations**
- Keep the language simple: Do **not overload** existing GAMS notation
- Use **existing language features** to specify additional model features, structure, and semantics
- Express **extended model information** in **symbolic (source) form** and apply existing modeling/solution technology
- Package new tools with the production system

GAMS

Stochastic Programming in GAMS

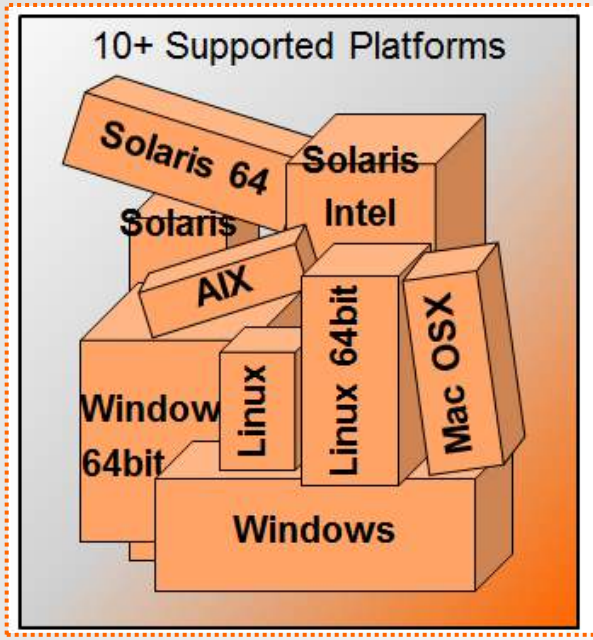
EMP/SP

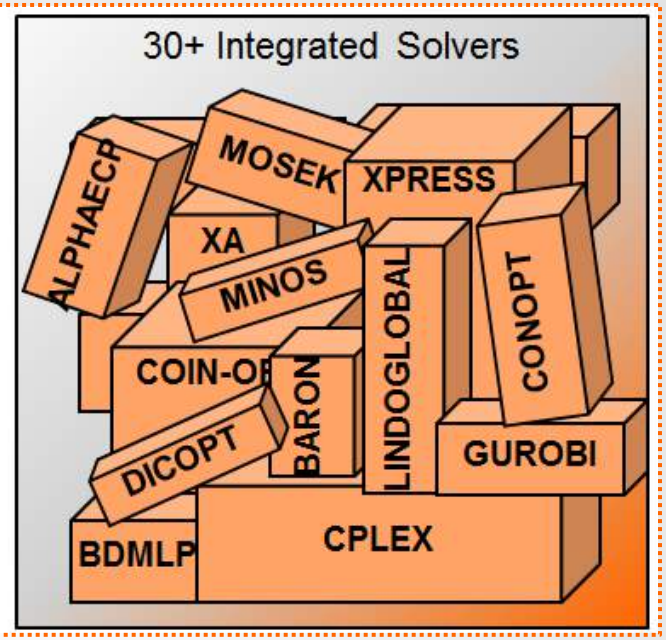
- Simple interface to add uncertainty to existing deterministic models
- (EMP) Keywords to describe uncertainty include: discrete and parametric random variables, stages, chance constraints, Value at Risk, ...
- Available solution methods:
  - Automatic generation of **D**eterministic **E**quivalent (can be solved with any solver)
  - Specialized commercial algorithms (DECIS, LINDO)



GAMS

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?
- …

GAMS

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 a	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	

GAMS

Why GAMS?

- Experience of 25+ years
- Broad user community from different areas
- Lots of model templates
- Strong development interface

- Consistent implemenation 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

Agenda

What is GAMS?

A Simple Example

Some Applications

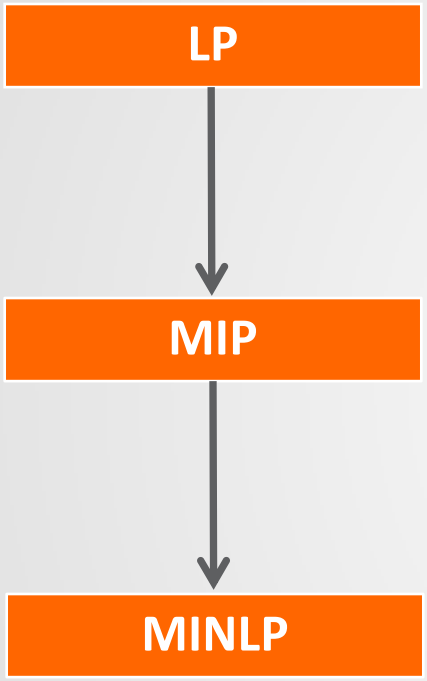
GAMS

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



- Determine minimum transportation cost.
  Result: city to city shipment volumes.
- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.
- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

GAMS

A Simple Transportation Problem

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

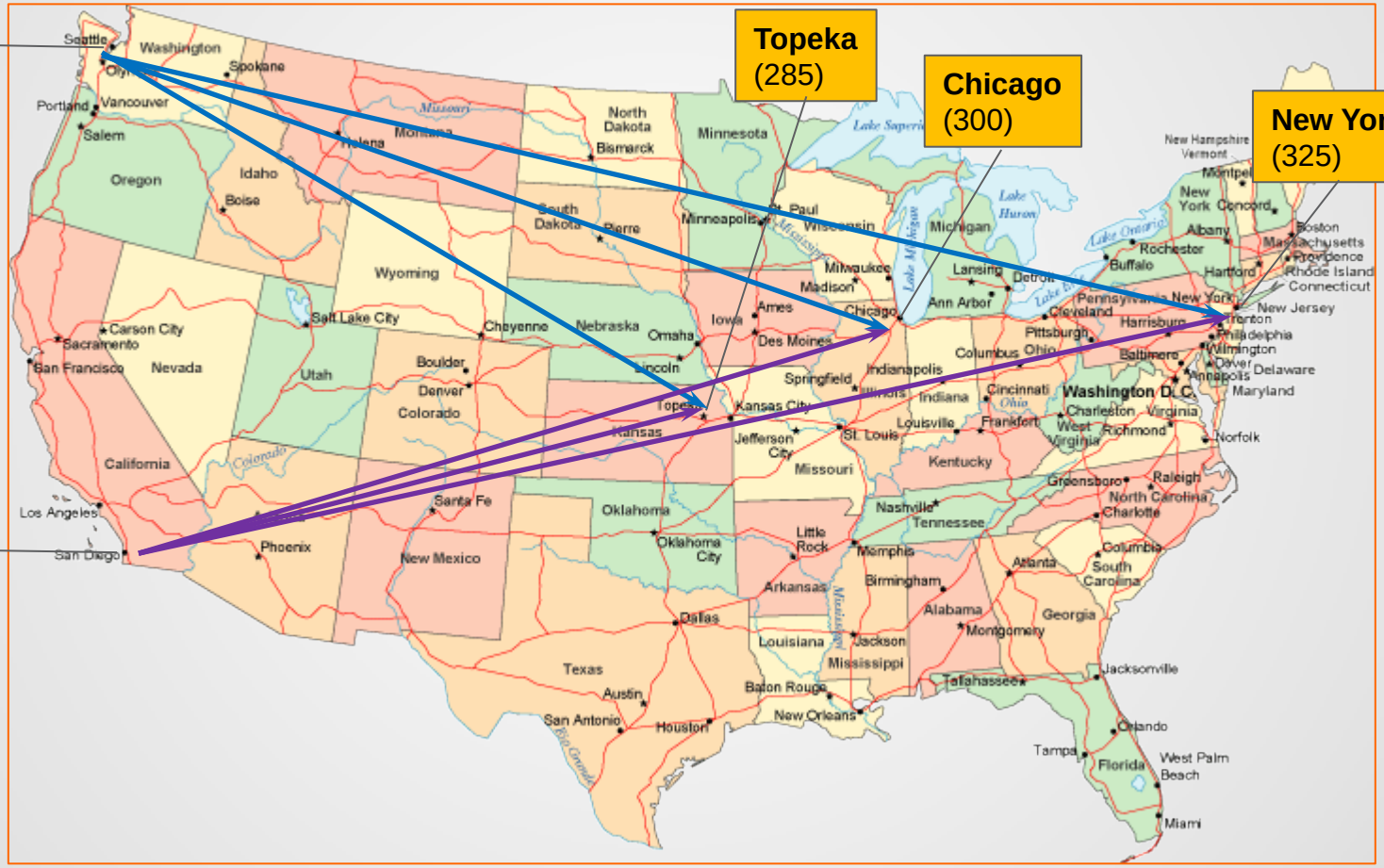
Seattle (350)

San Diego (600)

Topeka (285)

Chicago (300)

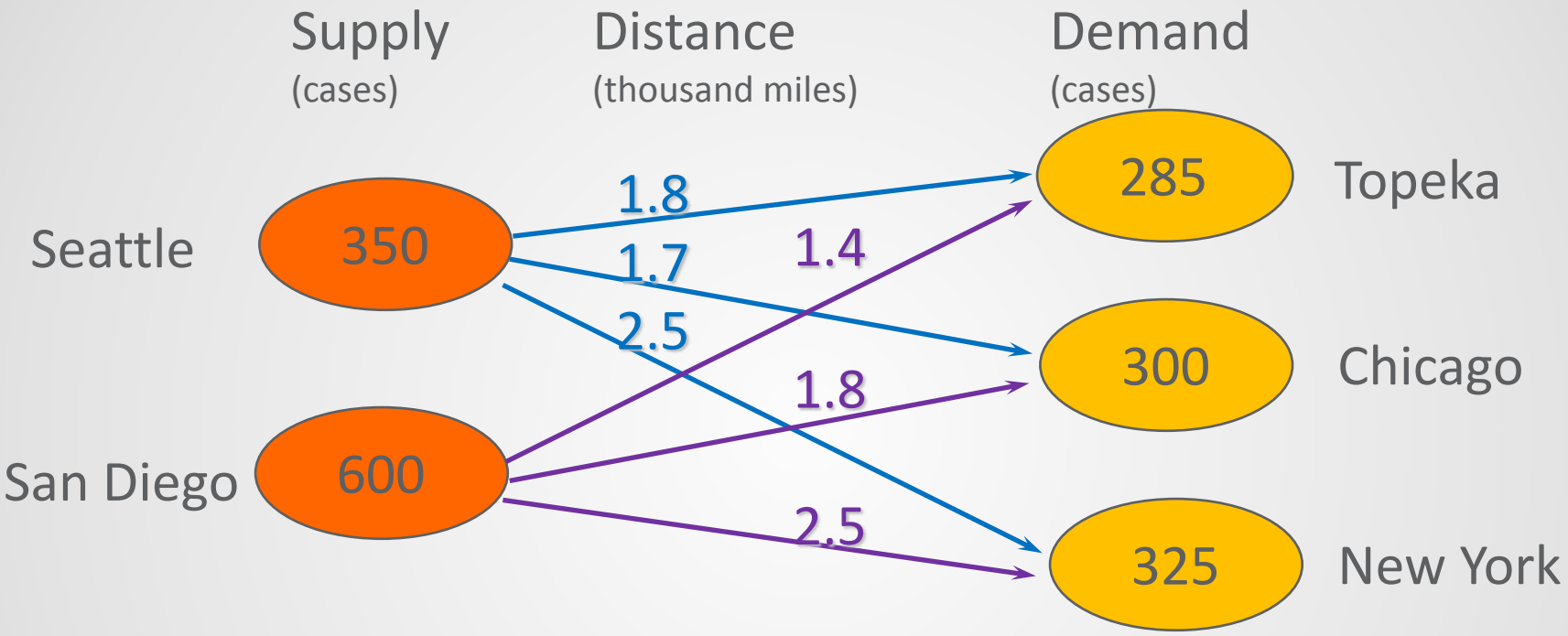
New York (325)



Freight: $90 case / thousand miles

GAMS

A Transportation Model



Minimize	Transportation cost
subject to	Demand satisfaction at markets
	Supply constraints

GAMS

Mathematical Model Formulation

Indices: i (Canning plants)

j (Markets)

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

Parameter: c_{ij} (Transport cost per case)

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

subject to

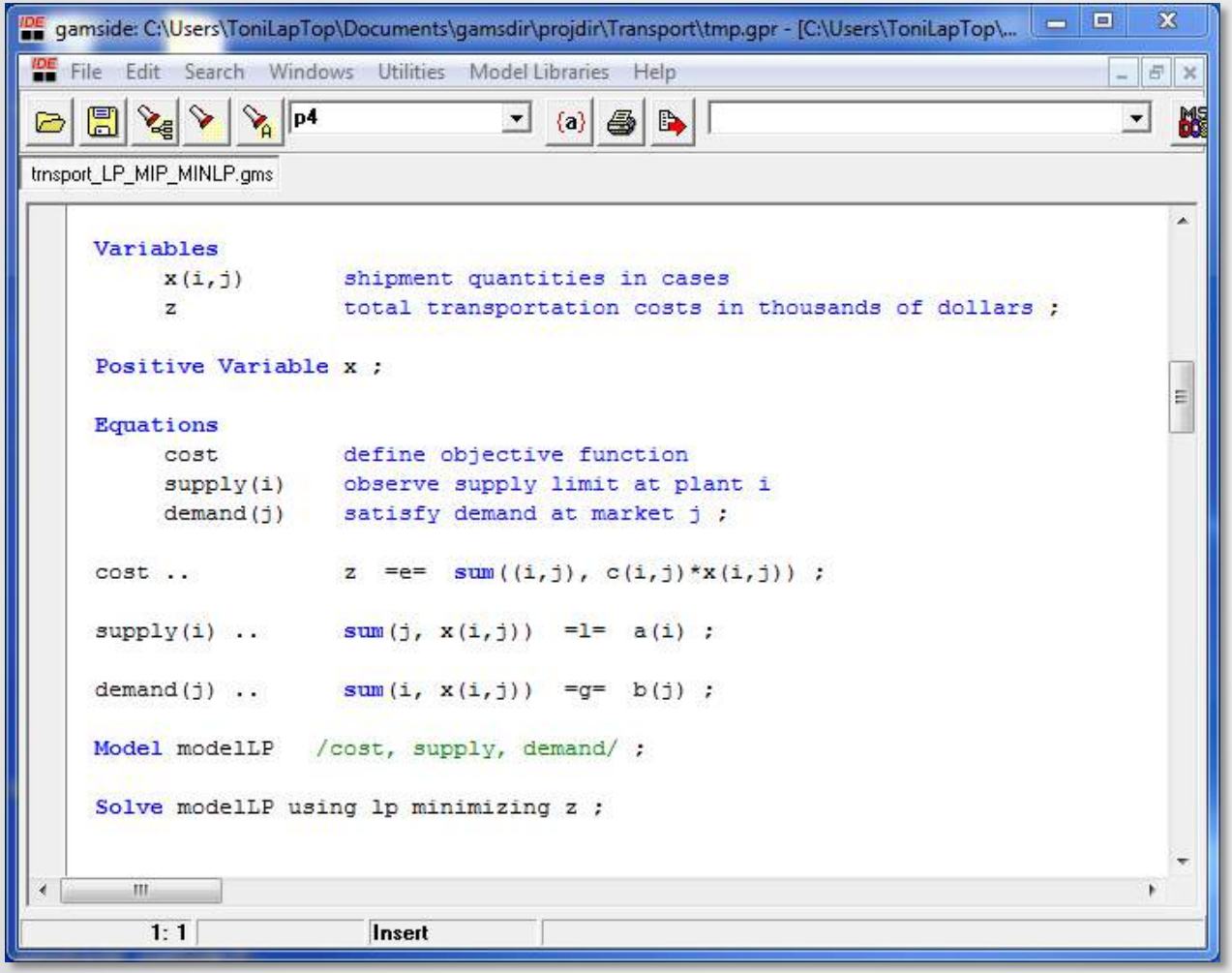
$\sum_j x_{ij} \leq a_i \qquad \forall\, i$ (Shipments from each plant $\leq$ supply capacity)

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

$x_{ij} \geq 0 \qquad \forall i, j$

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

GAMS Algebra (LP Model)



GAMS

Solution to LP model

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

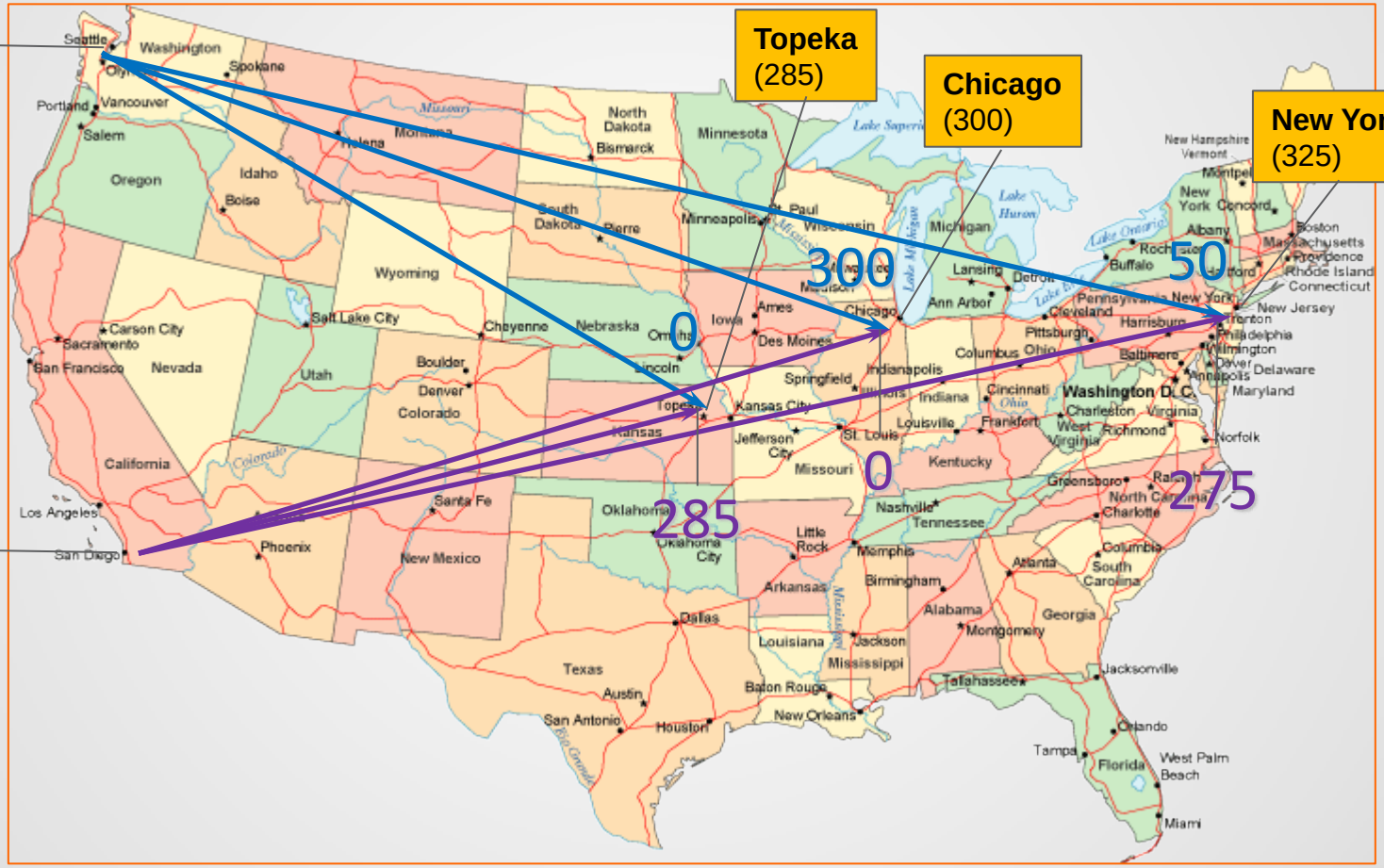
Seattle (350)

San Diego (600)

Topeka (285)

Chicago (300)

New York (325)

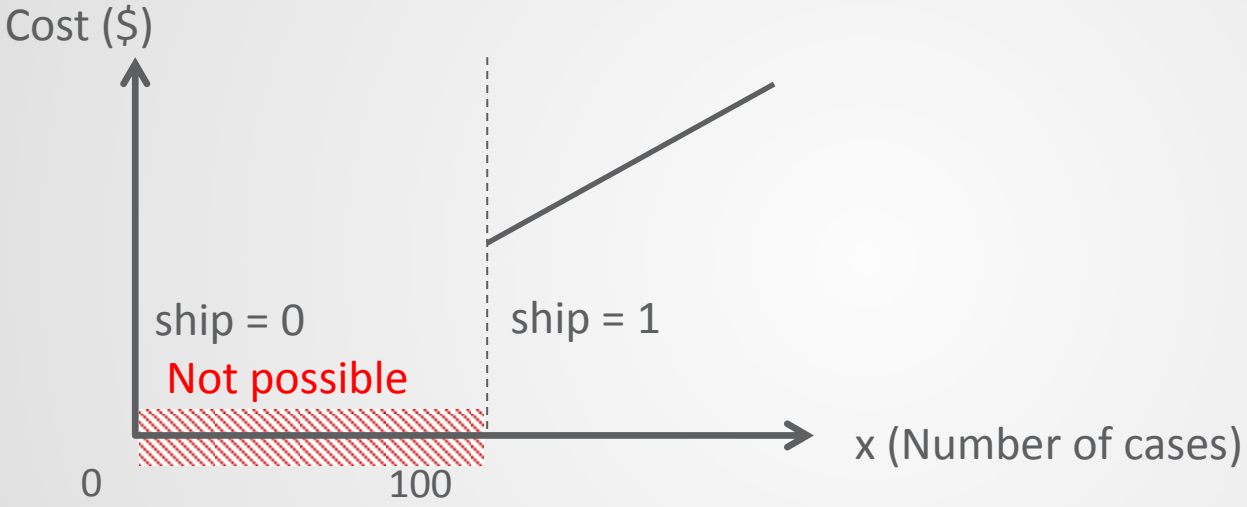


Freight: $90 case / thousand miles

Total cost: $154,935

Minimum shipment of 100 cases

- Shipment volume: **x** (continuous variable)
- Discrete decision: **ship** (binary variable, 0 or 1)



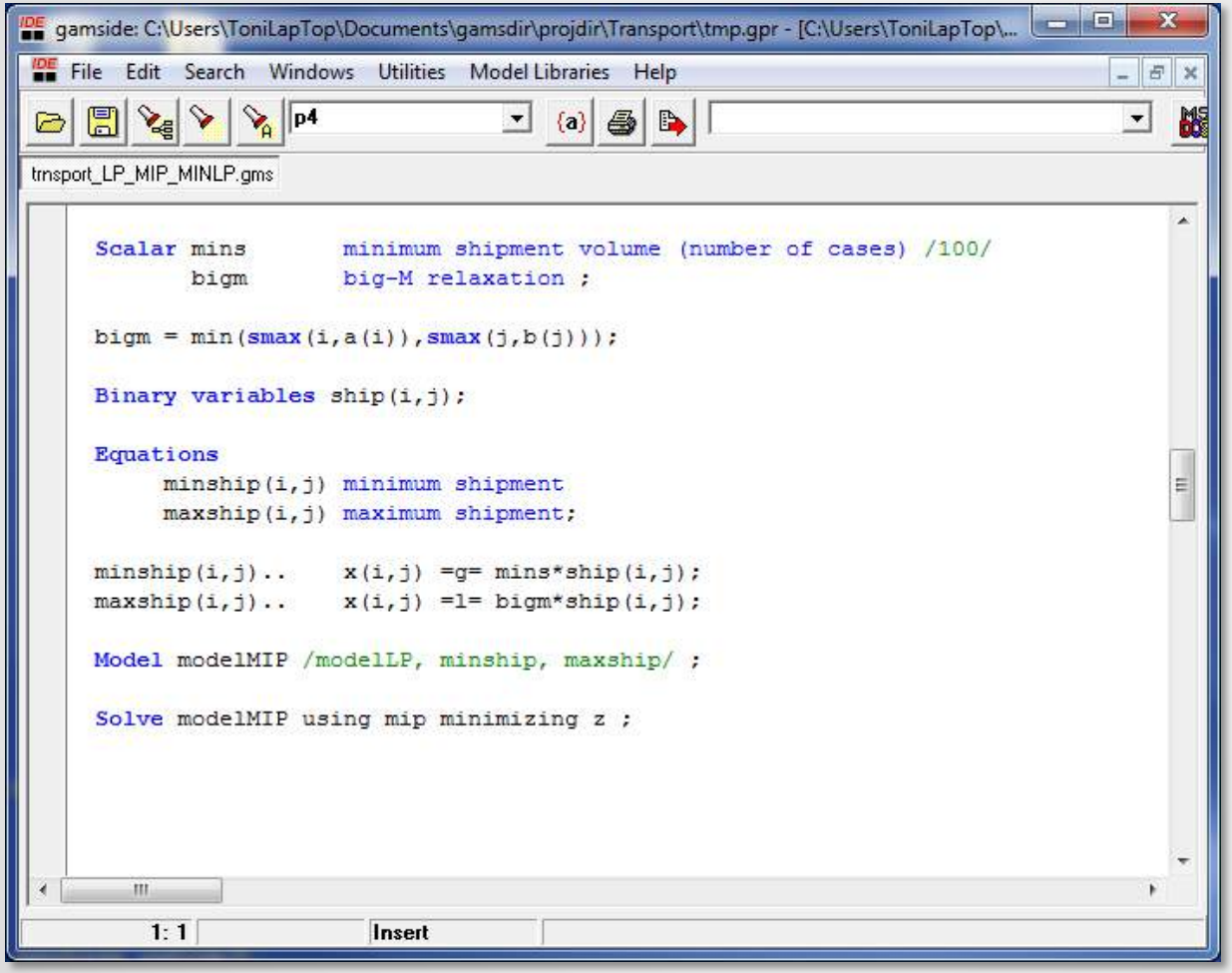
add constraints:

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

$x_{i,j} \leq 325 \cdot ship_{i,j} \quad \forall\, i,j$ (if ship=0, then do not ship at all)

$ship_{i,j} \in \{0,1\}$

GAMS Algebra (MIP Model)



GAMS

Solution to MIP model

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

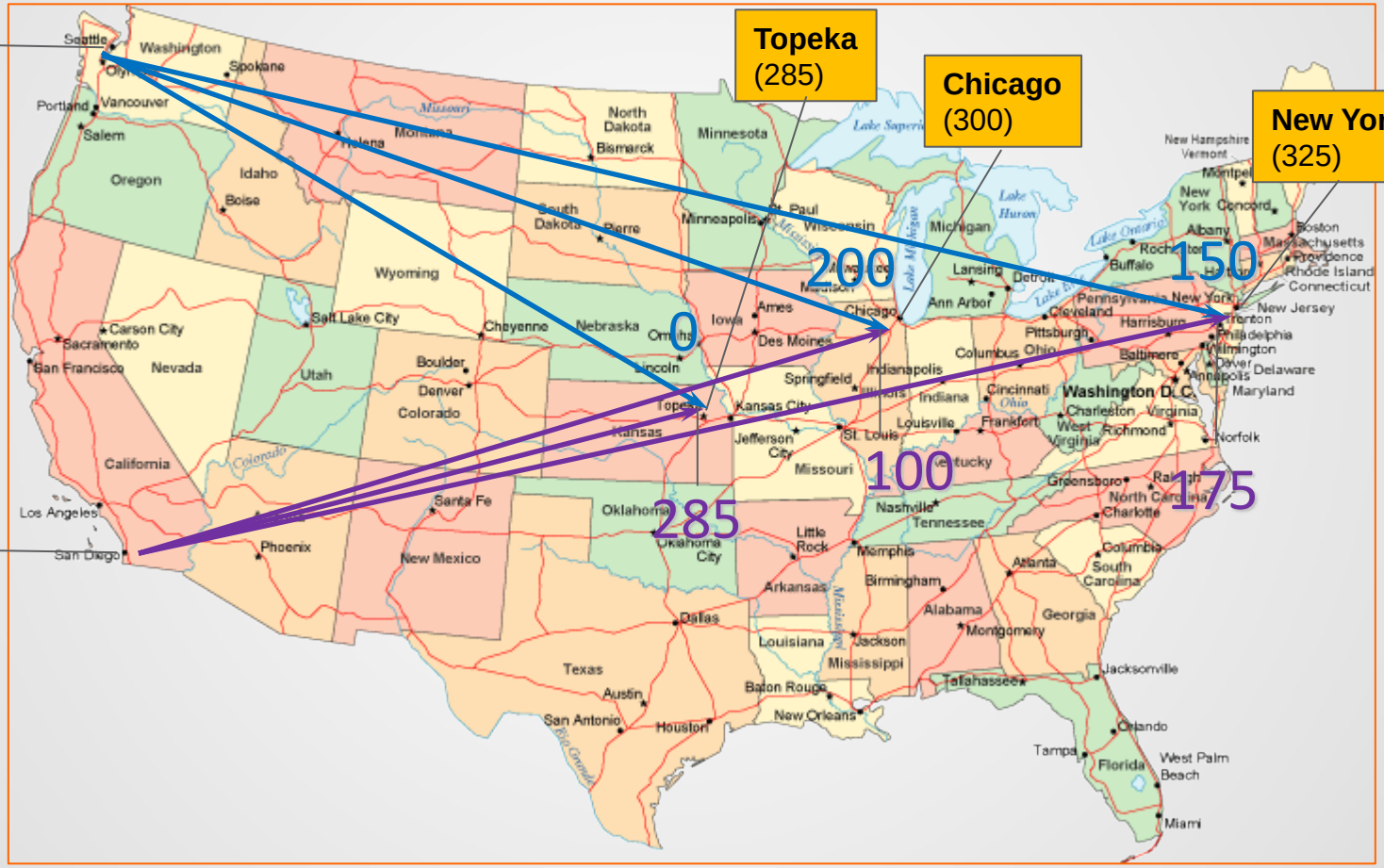
Seattle (350)

San Diego (600)

Topeka (285)

Chicago (300)

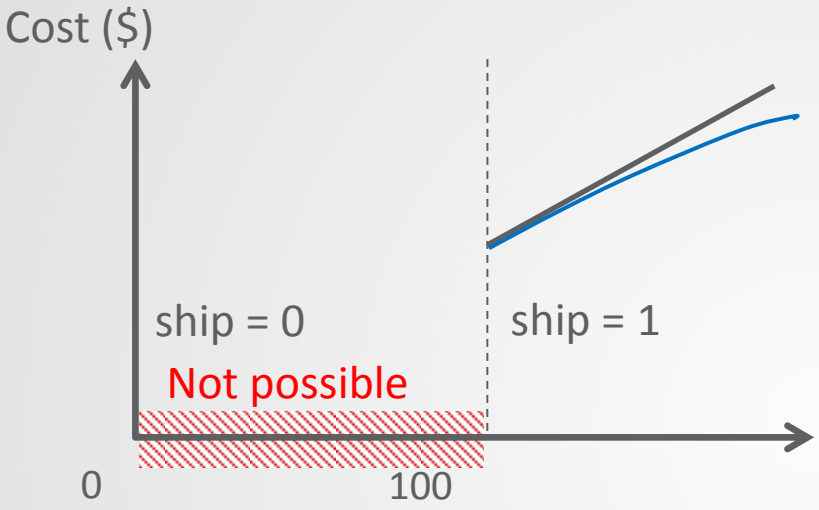
New York (325)



Freight: $90 case / thousand miles

Total cost: $155,835

Cost Savings (non-linear)



The cost per case decreases with a increasing shipment volume

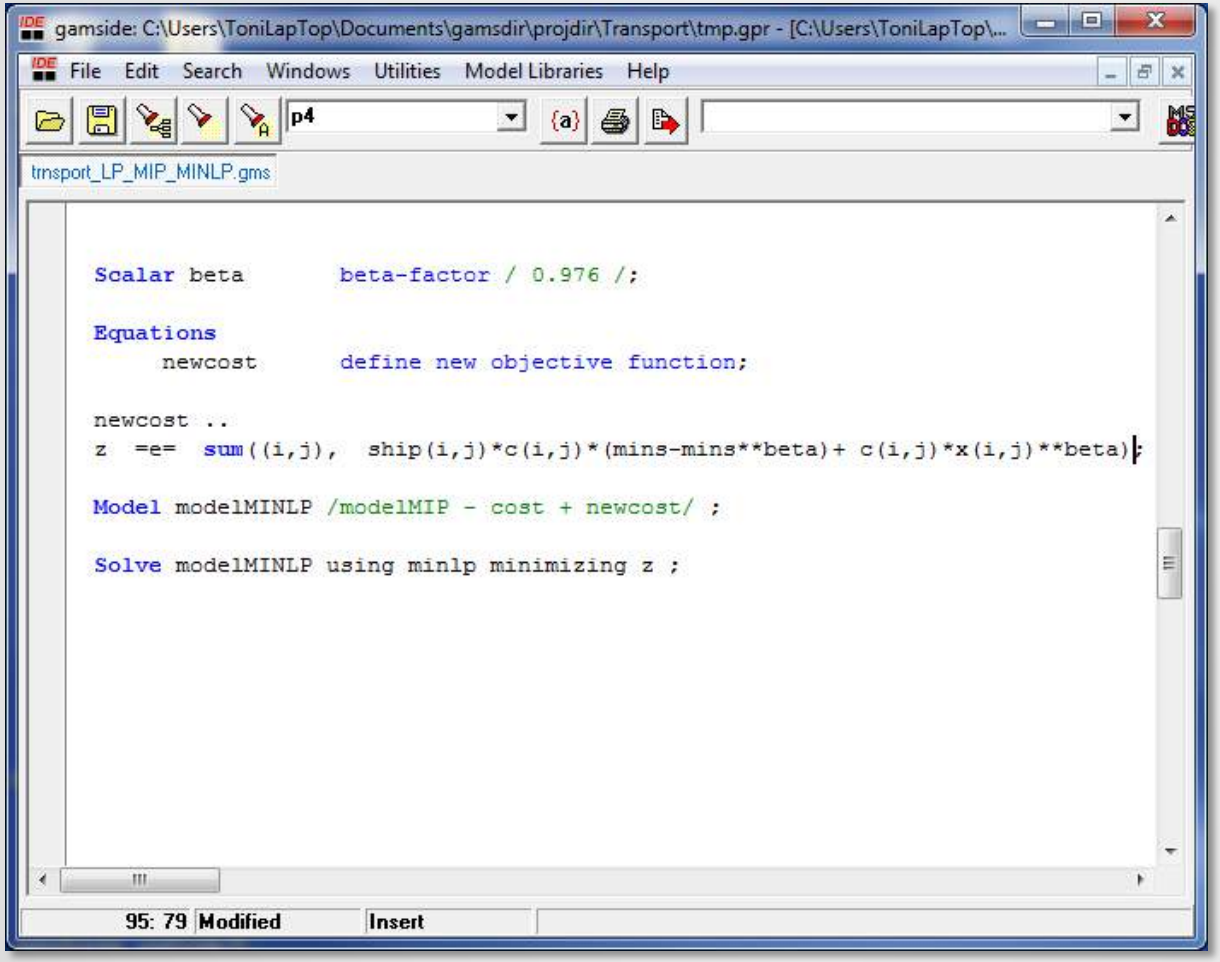
Replace:

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

With

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

GAMS Algebra (MINLP Model)



GAMS

Solution to MINLP model

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

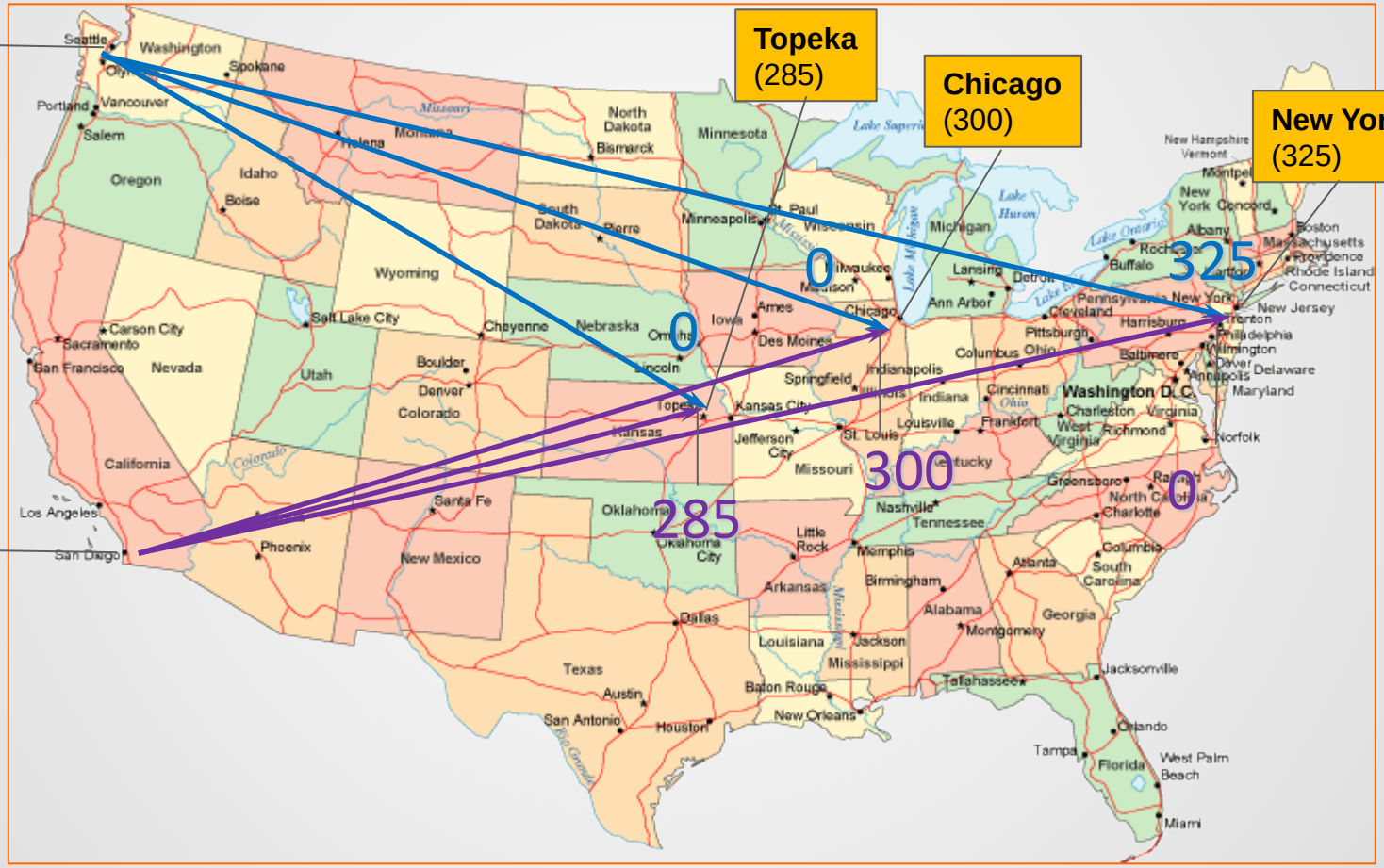
Seattle (350)

San Diego (600)

Topeka (285)

Chicago (300)

New York (325)



Freight: ~$90 case / thousand miles

Total cost: $142,752

Agenda Breakdown

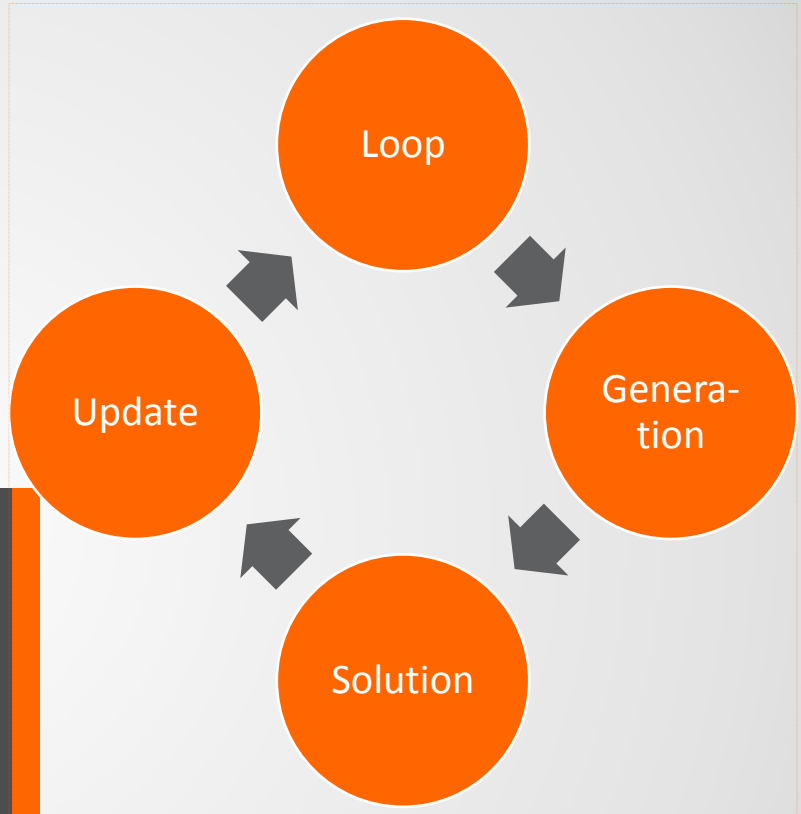
Some Applications

Scenario Solving with GAMS

- XYZ – Energy Company
- ACSOM

Outlook: GAMS under ODM Enterprise

Scenario Solving with GAMS



Problem: "small" number of independent scenarios…

- Through procedural language elements (e.g. loop statement)
- Works with any model type and solver
- Communication with solver through files or memory (in-core)

GAMS

Scenario Solver

…10,000's of independent scenarios…

- GAMS Scenario Solver allows to generate model once and updates the algebraic model keeping the model "hot" inside the solver.
- Platform independent, works with all solvers
- Performance close to native solver API
- Example: Stochastic model, 66,320 linear problems

Setting	Solve time (secs)	Factor
Loop: Solvelink=%Solvelink.Chainscript (default)	7,204	
Loop: Solvelink=%Solvelink.LoadLibrary%	2,481	18.3
GAMS Scenario Solver	392	
Cplex Concert Technology	210	1.86

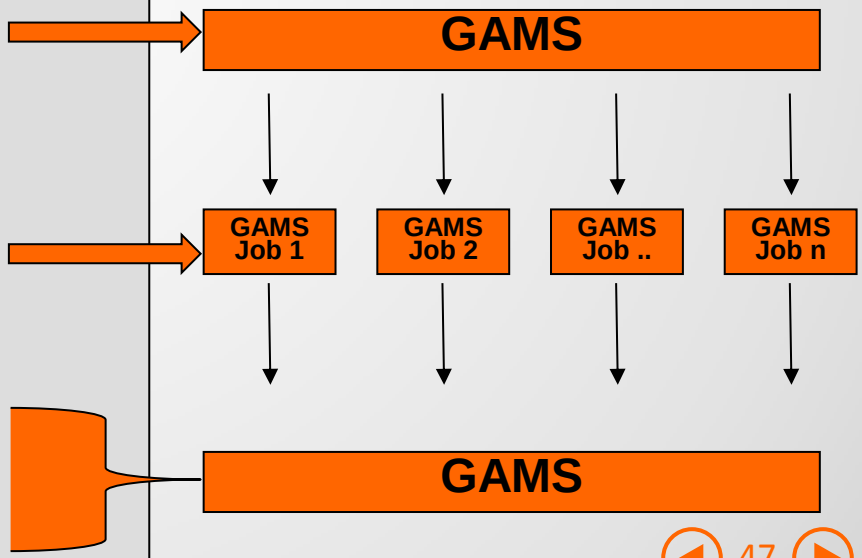
GAMS

GAMS GRID Facility

Running scenarios in a **distributed** environment…

- Scalable: support large grids, but also works on desktops
- Platform independent, works with all solvers/model types
- Only minor changes to model required

1. Submission of jobs
2. "Grid Middleware"
   - Distribution of jobs
   - Job execution
3. Collection of solutions
4. Processing of results



GAMS

Massive Parallel in the Cloud

XYZ- Energy Company

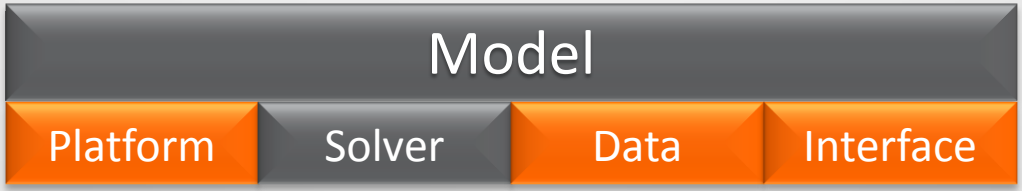
- Large scale MIP model
- Failures with Stochastic Optimization and Robust Optimization
- Scenario Analysis works!
- Task: Solve 1000 scenarios (MIPs,1-2 hours) every week overnight
- Security issues: Obfuscation of (binary) GAMS files
- Post processing of results outside modeling system

GAMS

Massive Parallel in the Cloud

amazon webservices

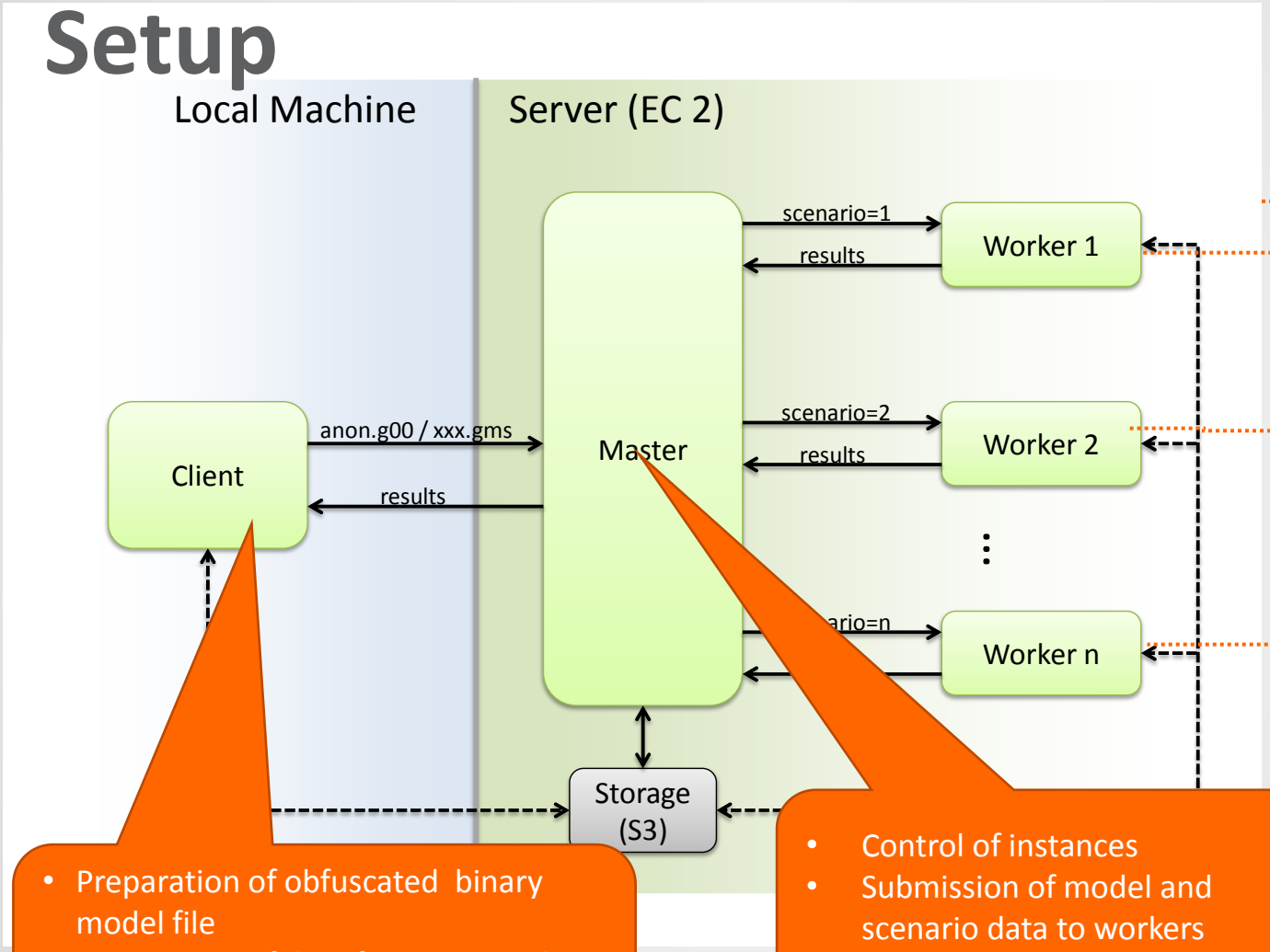
- Deployment environment: „Cloud“ (Amazon EC2)
- A few hundred parallel instances (workers)
- Automated setup, including
  - Start instances
  - Prepare / Submit / Run GAMS jobs
  - Collect results
  - Stop instances
- Python
  - Boto: Python interface to Amazon webservices
  - Setup and control of instances (Workers)
  - GAMS OO API (Python): Control of GAMS jobs



GAMS

Massive Parallel in the Cloud

Setup



amazon web services™

Configured AWS Instances
- Python
- GAMS

- Preparation of obfuscated binary model file
- Preparation of data for every worker
- Mapping of results into original namespace

- Control of instances
- Submission of model and scenario data to workers
- Collection and merge of results

GAMS

ACSOM

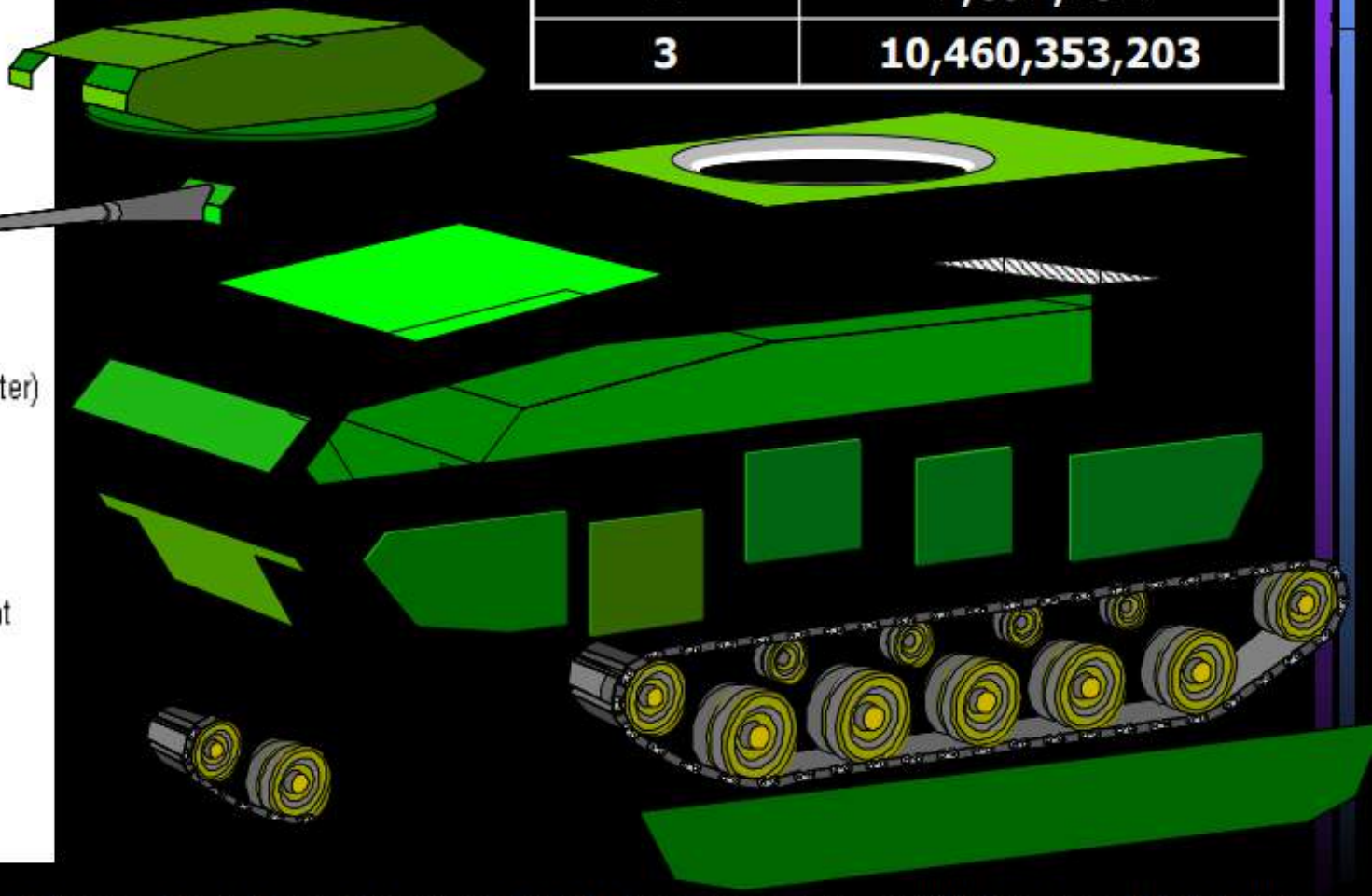
- **A**dvanced **C**ollaborative **S**ystem **O**ptimization **M**odeler
- Configuration of (expensive) military vehicles
- Project partners:
  - General Dynamics Land Systems (GDLS)
  - Industrial & Systems Engineering, Wayne State University
  - GAMS Development Corp
  - Amsterdam Optimization Group

The Whole-system Design Problem

21 SUBSYSTEMS

- AFES
- Armor
- Chassis Structure
- Crew Station
- Defensive Armament
- Dismountable
- ECS
- Fuel
- Hit Avoidance
- Lighting
- Mission Equipment (Shooter)
- Mission Station
- Mission Structure
- NBC
- Platform Electronics
- Power Distribution & Mgmt
- Propulsion
- Signature Management
- Suspension
- Turret Structure
- Water Management

Options per subsystem	Theoretically Possible Subsystem Combinations
2	2,092,152
3	10,460,353,203



GAMS

Project Scope

Reimplementation of ACSOM 1.2 algorithm

Significant performance improvements

Use multiple cores

- For current setup (laptops)
- For high-performance machines
- Be prepared for advancing technologies (more parallelism)

GAMS

Scenario Solver and Parallel Combined

Implementation	Number of MIP models	Solve time	Rest of algorithm	Total time
Traditional GAMS loop (call solver as DLL)	100,000	1068 sec	169 sec	1237 sec
Scenario Solver	100,000	293 sec	166 sec	459 sec

Implementation	Number of MIP models	Worker Threads	Parallel sub-problem time	Rest of algorithm (serial)	Total time
Parallel + Scenario Solver	100,000	4	116 sec	67 sec	183 sec

http://yetanothermathprogrammingconsultant.blogspot.de/2012/04/parallel-gams-jobs-2.html

GAMS

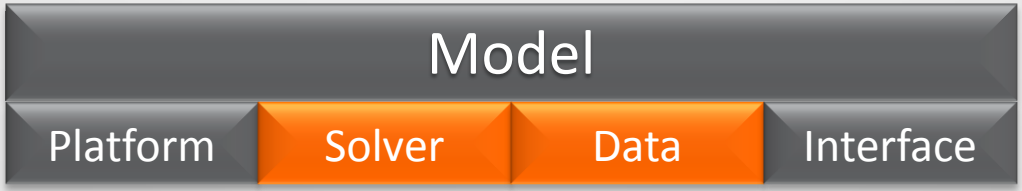
ACSOM Project Summary

New System

- GAMS (complete algorithm; streamlined design)
- Multiple parallel instances of scenario solver
- Optimized communication with database

Performance improvement (same hardware)

- From **hours** and even **days** for some instances
- To **minutes**, up to an **hour**



Scenario Solving with GAMS

Various options:

- Loop (declarative language elements)
- Scenario Solver
- Grid Facility
- Parallel Execution

Simple extensions of Model:

- Independence of Model and Platform
- Independence of Model and Solver
- Independence of Model and Interface

Agenda Breakdown

Some Applications

Scenario Solving with GAMS

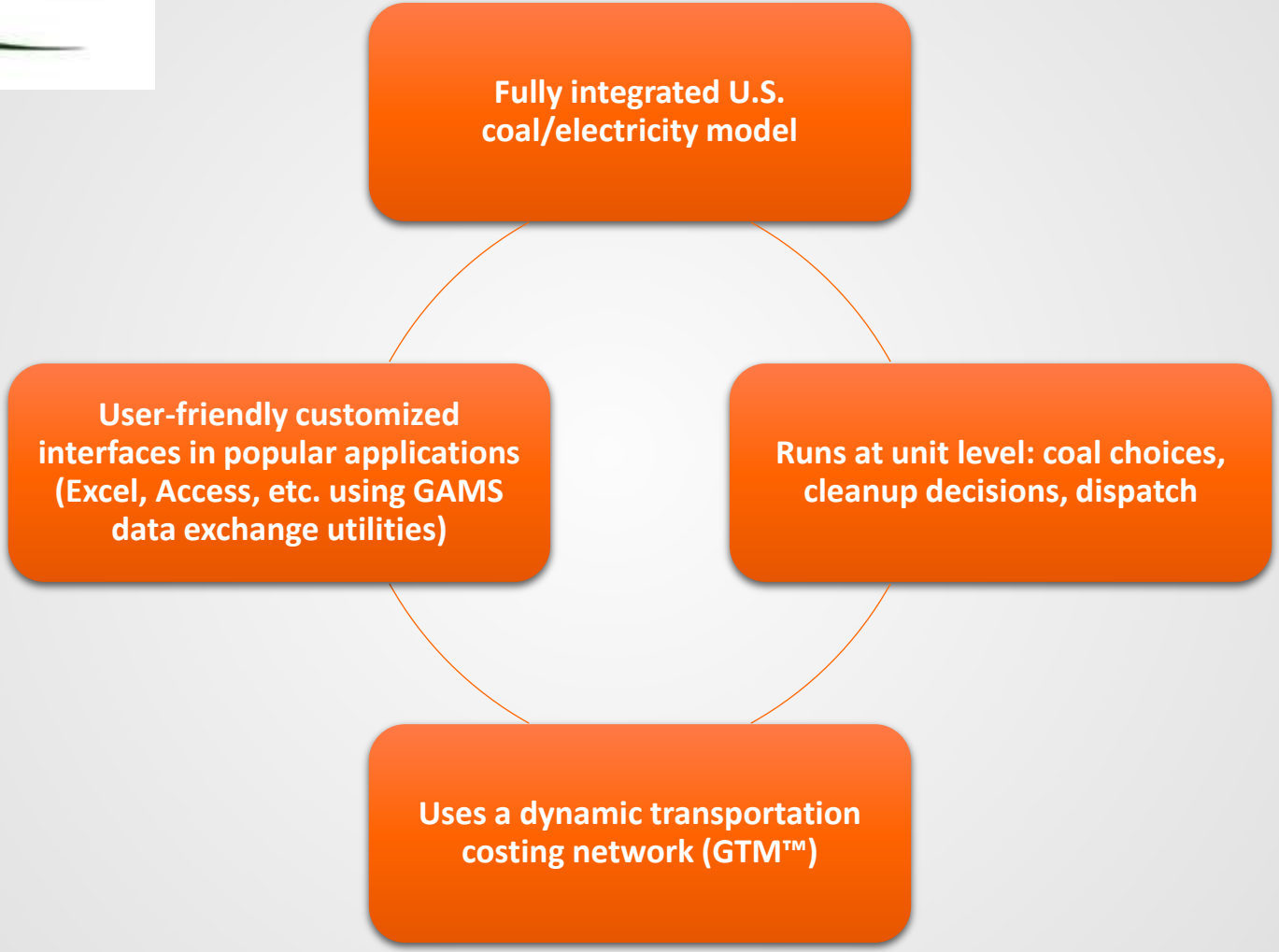
- XYZ – Energy Company
- ACSOM

Outlook: GAMS under ODM Enterprise

GAMS

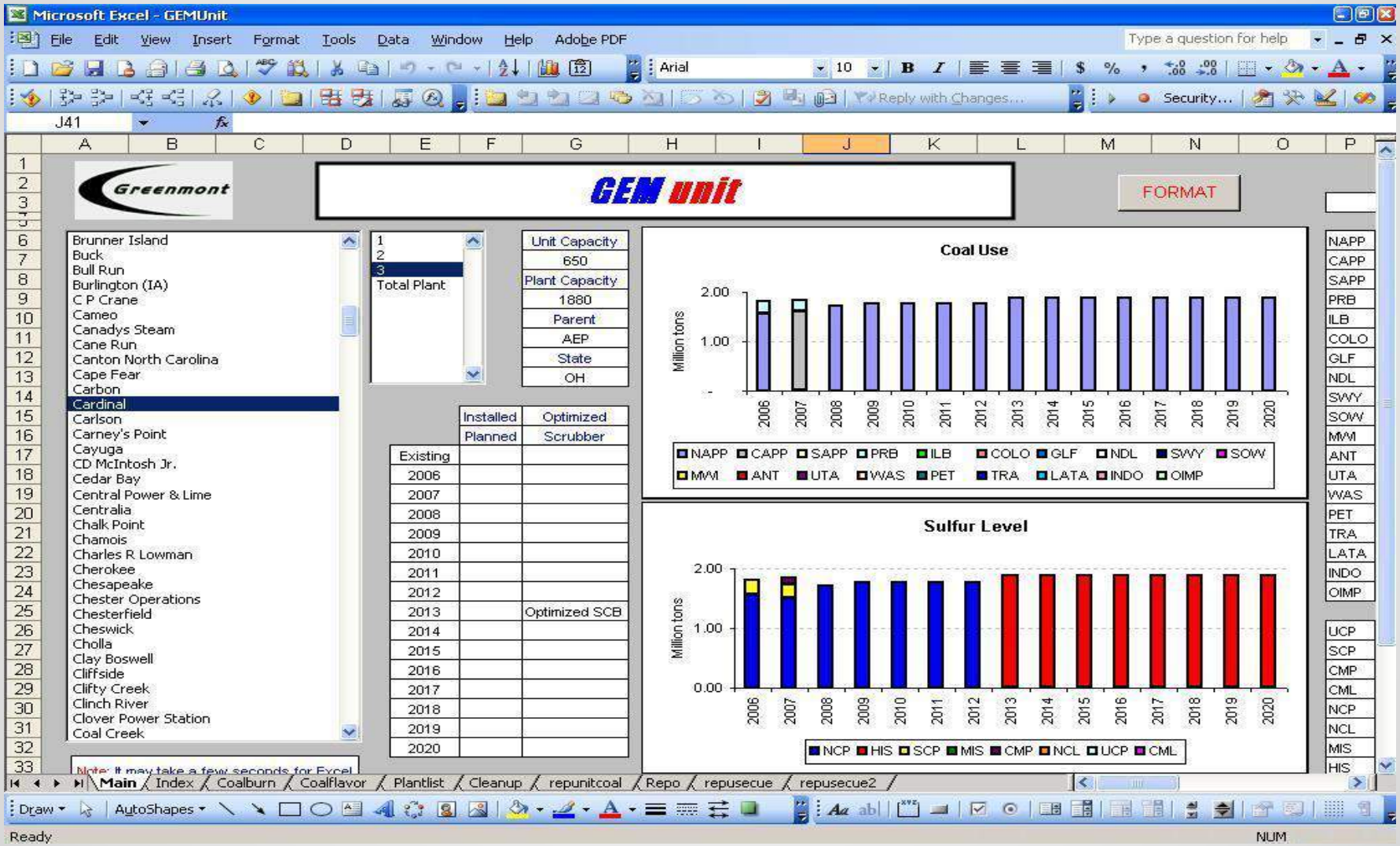
Greenmont

GEM Energy Models



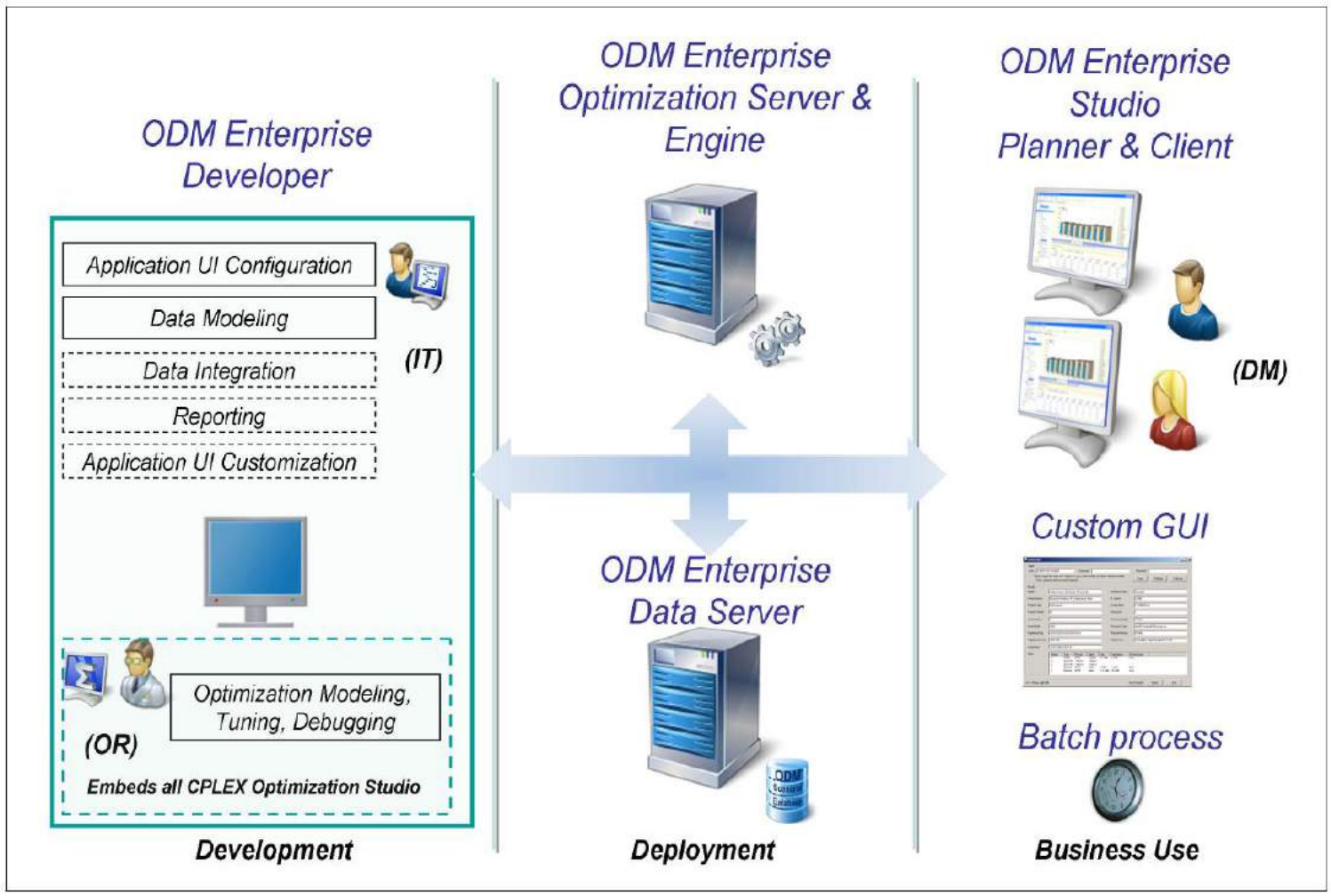
GAMS

GEM Excel Interface



GAMS

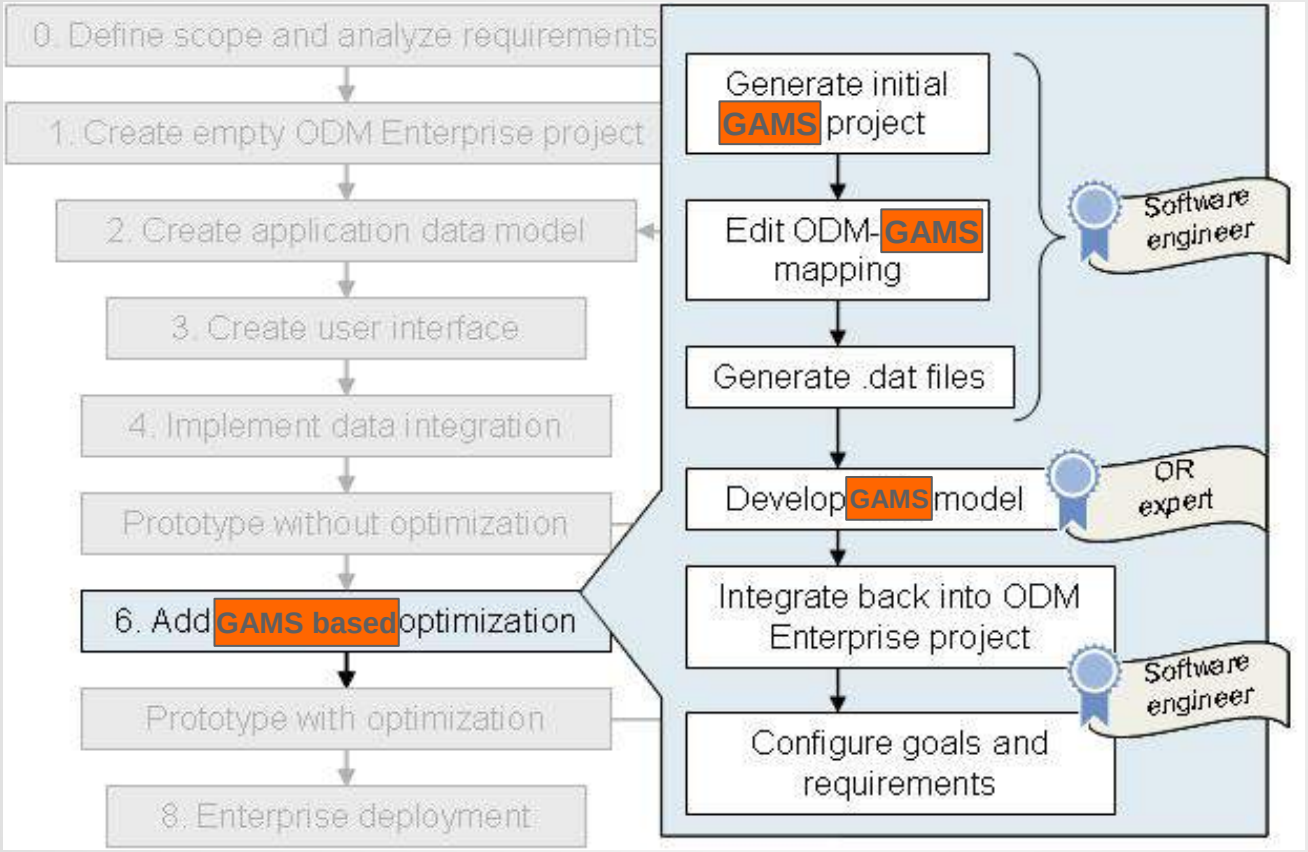
ODM System Architecture



IBM

GAMS

Replacing OPL with GAMS

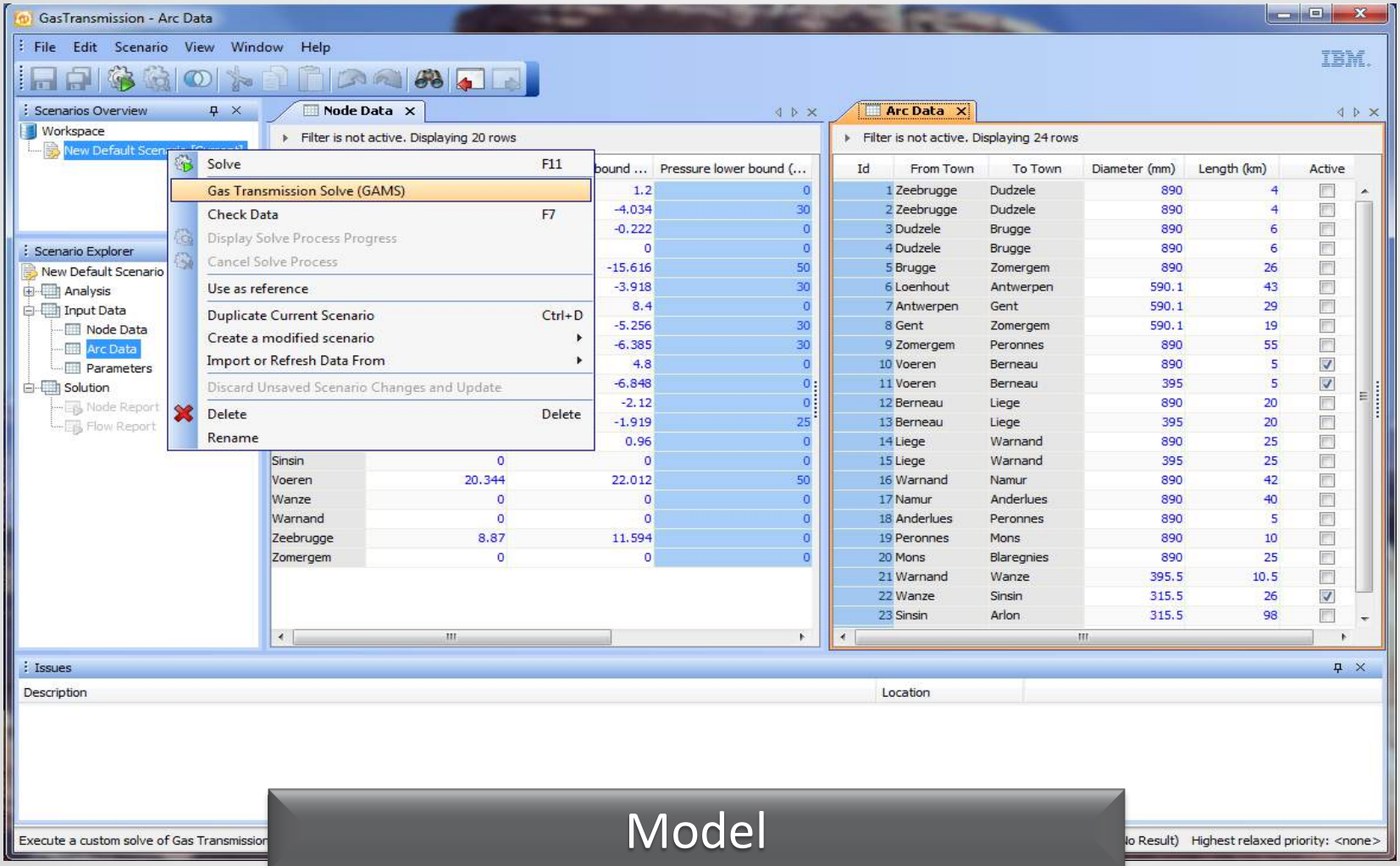


Implementation:

- ODM Custom Task (GAMS Solve Task)
- GAMS OO Java – API (Data Exchange, Job Control)

GAMS

GAMS Gas Transmission Model in ODM



Model

Platform | Solver | Data | Interface

Summary

GAMS Design Principles

- Simple language
- Open architecture with independent layers

Striving for Innovation and Compatibility

- Models benefit from new developments
- Protect investments of users

Don‘t lock users into a certain environment!

Thank You

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

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