Pre-Conference Workshops

Michael Bussieck
Steve Dirkse
Fred Fiand
Lutz Westermann

Outline

An Introduction to GAMS: Agenda

Company

- *Roots: World Bank, 1978 – 1987 Initial product*
- Went commercial in 1987
- GAMS Development Corp. (USA), GAMS Software GmbH (Germany)
- 2016: New management team
- Software Tool Provider

GAMS at a Glance

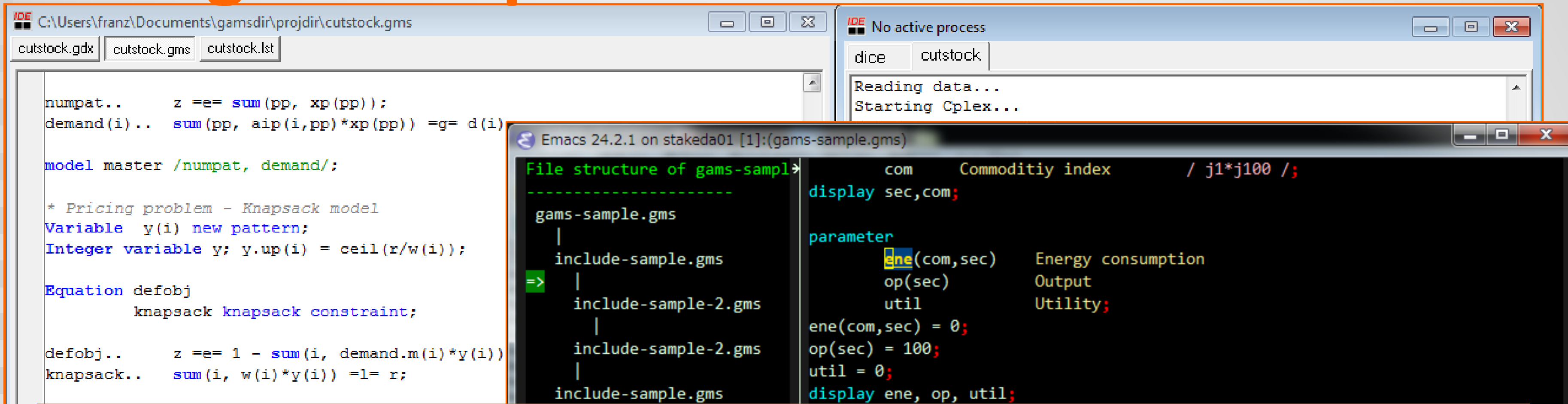
- Pioneered Algebraic Modeling Languages
- Robust, scalable state-of-the-art algebraic modeling technology for complex, large-scale optimization
- Open architecture and uniform interface to all major commercial and academic solvers (30+ integrated)
- Evolution through more than 25 years of R&D and user feedback, maturity through experience and rigorous testing

GAMS Users and Application Areas

- 13,500+ licenses
- Users: 50% academic, 50% commercial or governmental
- Used in more than 120 countries
- 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

Strong Development Environment



- Project management
- Editor / Syntax coloring / Spell checks
- Model Debugging / Profiling
- Solver selection / Option selection
- Data viewer
- GAMS Processes Control

Uniform System Documentation

GAMS | Documentation | Model Libraries | Search | Index | Help

- GAMS Documentation 24.9
  - Release Notes
  - Installation Notes
  - Licensing
  - Tutorials and Examples
  - User's Guide
  - Solvers
  - APIs
  - Tools
  - Glossary

GAMS Documentation 24.9

GAMS Documentation Center

The GAMS Documentation Center provides you with the technical information on getting started, using and maintaining our GAMS (General Algebraic Modeling System) products.

- Release Notes - What's new in GAMS and all chang
- Installation Notes - GAMS Installation guides on diffe
- Licensing - GAMS Licensing
- Tutorials and Examples - Step-by-step guides includ
- User's Guide - Guides through GAMS Language and

GAMS | Documentation | Model Libraries | Search | Index | Help

- Licensing
- Tutorials and Examples
- User's Guide
  - Introduction
  - GAMS Programs
  - Set Definition
  - Dynamic Sets
  - Sets as Sequences: O
  - Data Entry: Parameter
  - Data Manipulations wi
  - Variables
  - Equations
  - Model and Solve State
  - Conditional Expressior
  - The Display Statemen
  - Programming Flow Co
  - The Grid and Multi-Thr
  - Special Language Fea

This documentation guides GAMS User through several topics in GAMS system.

- **GAMS Language** - This part introduces the components of the GAMS language in an ordered way, interspersed with detailed examples that are often drawn from the model library. All models from the model library are enclosed in square parenthesis (for example, [TRNSPORT]).
  - **Introduction** - an introductory to GAMS User's Guide.
  - **GAMS Programs** - The structure of the GAMS language and its components
  - **Set Definition** - The declaration and initialization of sets, subsets, and domain checking.
  - **Dynamic Sets** - The membership assignment, the usage of dollar controls, and set operations.
  - **Sets as Sequences: Ordered Sets** - Special features used to deal with a set as if it were a sequence.
  - **Data Manipulations with Parameters** - The declaration and assignment of GAMS parameters.
  - **Data Entry: Parameters, Scalars and Tables** - Three basic forms of GAMS data types : Parameters, Scalars and Tables.

GAMS | Documentation | Model Libraries

Model Libraries

The Model Libraries contains a large number of GAMS Models includi

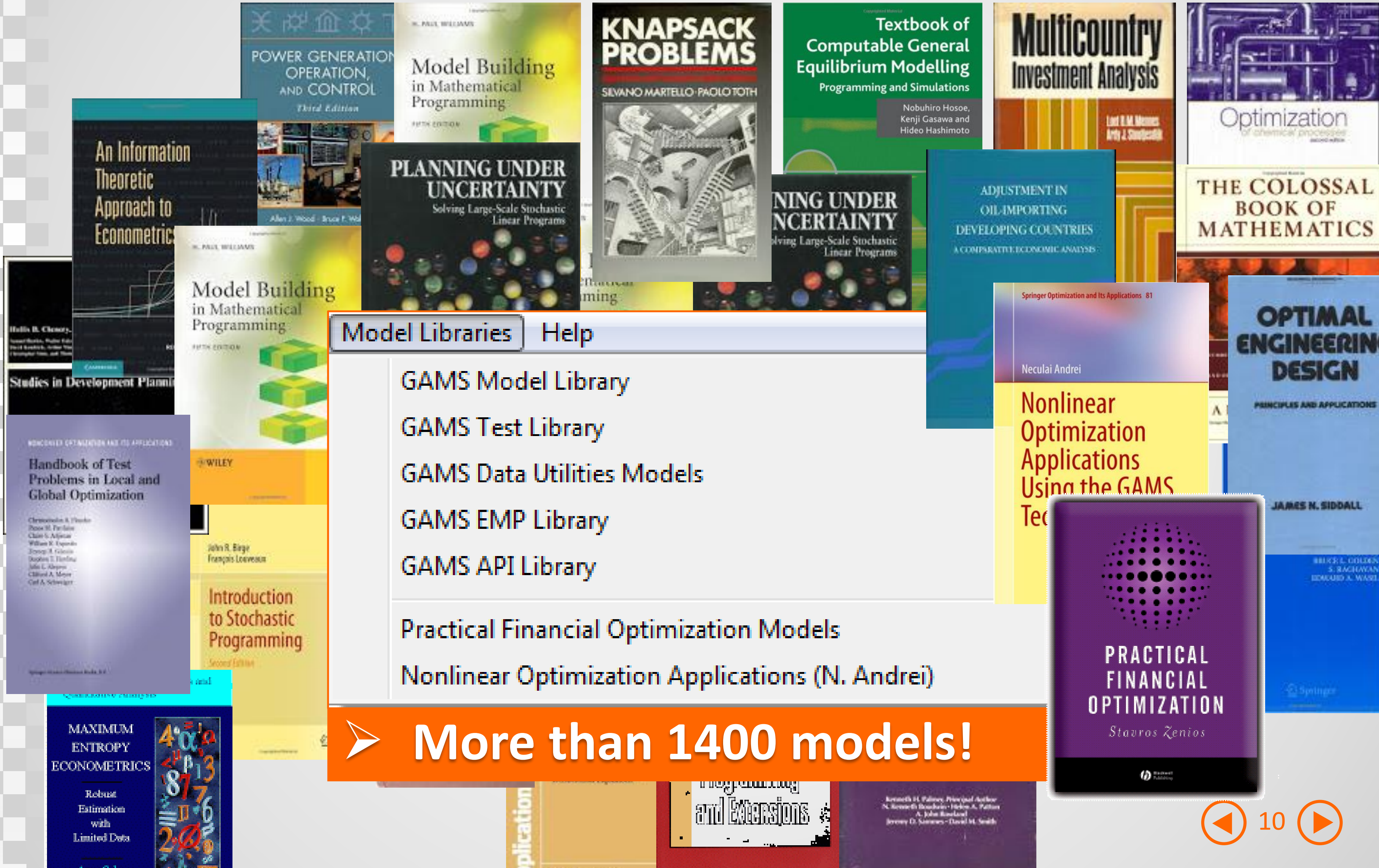
- GAMS Model Library - GAMS models representing interesting a shipment by firms, investment planning, cropping patterns in ag macroeconomics stabilization, applied general equilibrium, inte networks, and many more.
- GAMS Test Library - GAMS models developed for testing and many solvers distributed with the GAMS system.
- GAMS Data Library - GAMS models demonstrating various utili as spreadsheets and database interface.
- GAMS EMP Library - GAMS Extended Mathematical Programn GAMS/EMP.
- GAMS API Library - GAMS Models used as scripts to compile a languages interfacing to GAMS.
- FIN Library - GAMS practical financial optimization models des Making for Financial Engineers by Consiglio, Nielsen and Zenic
- NOA Library - GAMS nonlinear optimization applications model

Simple Integration of GAMS Models

Object Oriented API's

- Use GAMS for modeling and optimization tasks
- Connects GAMS to other environments
  - Programming languages (.Net, C++,Java, Python
  - Applications (through Smart Links)
  - (New) Embedded Code (Python)
- Communication through Memory or Files

Free Model Libraries



➢ **More than 1400 models!**

Where to Find Help?

- *Documentation Center: https://www.gams.com/latest/docs/*
- *Free Model Libraries: https://www.gams.com/latest/docs/modlibsindex.html*
- Mailing Lists, Newsletters, and Forum:
  - https://www.gams.com/community/newsletters-mailing-list/
  - //forum.gamsworld.org/
- YouTube Channel: https://www.youtube.com/user/GAMSLessons
- **GAMS Support: support@gams.com**

Foundation of GAMS

Powerful algebraic modeling language

Open architecture with interfaces to other systems

Independent layers

Powerful Declarative Language

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

Mix of Declarative and Imperative Elements

Control Flow Statements (e.g. loops, for, if,…), macros and functions

Advantages:

- Build complex problem algorithms within GAMS
- Simplified interaction with other systems:
  - Data exchange
  - GAMS process control

Foundation of GAMS

Powerful algebraic modeling language

Open architecture with interfaces to other systems

Independent layers

Open Architecture

Designed to interact with other Systems

- Solver Interfaces: Proprietary or Open Source (COIN)
- Data Exchange and GAMS Control
  - ASCII or Binary (GDX)
  - OO-API (.Net, C++, Java, Python, file or memory)
  - Smart Links to other applications (e.g. MS Excel, Databases, Matlab, R,...)
- Code Embedding

Foundation of GAMS

Powerful algebraic modeling language

Open architecture with interfaces to other systems

Independent layers

Model

Platform	Solver	Data	Interface

Separation of Model and Platform

Supported Platforms

Microsoft Windows

Linux

Welcome to Mac OS X

solaris

AIX

- **Move models between platforms with ease!**

Model

Platform | Solver | Data | Interface

Separation of Model and Solver

One environment for a wide range of 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 · COIN|OR

➢ **More than 30 Solvers integrated!**

Model

Platform | Solver | Data | Interface

Separation of Model and Solver

Uniform interface to all major solvers

- Switching between solvers with one statement
- Unified Documentation
- Licensing (GAMS as a „license broker“)

Av. number of commercial solvers per license

- Academic clients: 2.9
- Commercial: 2.2

Separation of Model and Data

```
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 ;
Scalar f ;

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 transport /all/ ;
```

- Declarative Modeling
- Sparse Data Structures
- Various ways to exchange data
  - ASCII
  - Binary

GAMS

Separation of Model and User Interface

No preference for a particular user interface

- Open architecture and interfaces to other systems
  - OO-API's for seamless integration
  - Smart Links
- Mode of Operation
  - Interactive or Embedded / Batch
  - Local or Remote

Application – Cloud Computing

xyz Energy Company

Scenario Analysis in the Cloud

- Solve 1,000+ scenarios (MIPs, one hour) every week overnight
- Issues:
  - Costs (Licensing)
  - Automation / Security

Model

Platform | Solver | Data | Interface

GAMS

Application – Cloud Computing

xyz – Energy Company

Implementation:

- Amazon Cloud: 1,000+ parallel machines (instances), Python, GAMS + OO Python API
- Automated setup, including
  - Starting instances
  - Prepare / Submit / Run GAMS jobs
  - Collect results
  - Stop instances

Model

Platform | Solver | Data | Interface

Application – Cloud Computing

Commercial Aspects

"Hardware" Amazon Cloud (1,000 instances) :

Hardware Costs / run: **$70!**

(1,000 instances/run * $0.07 instance / hour)

Software Licensing:

- Gurobi and IBM offer per-usage license
- Client with strong preference for annual license fee, not a per-usage license

Application – Cloud Computing

45 Provided Model Instances

- Statistics:
  - 163,608 – 1,959,550 rows
  - 84,930 – 983,587 var. (32,240-258,796 dis.)
  - 447,537 – 6,068,729 NZ
- Tests with CPLEX, SCIP, and CBC
- 60 minutes, gap max. 1%
- Manual option tuning for SCIP (thanks to Gerald Gamrath & Ambros Gleisner)

Application – Cloud Computing

Results

- CPLEX: All instances solved to optimality
- SCIP:
  - Could solve all 45 instances
  - But: After 60 min. 2 instances with gap > 20%
- CBC:
  - Did also well
  - But: After 60 min. no solution for some instances (< 10%)

Application – Cloud Computing

Proposed Strategy

- Run all instances simultanuesly with SCIP and CBC

  → *„hardware" costs: $0,07 per instance hour*

- After 60 minutes take the best solution

- If necessary solve „difficult" model instances with CPLEX (outside the cloud)

Agenda

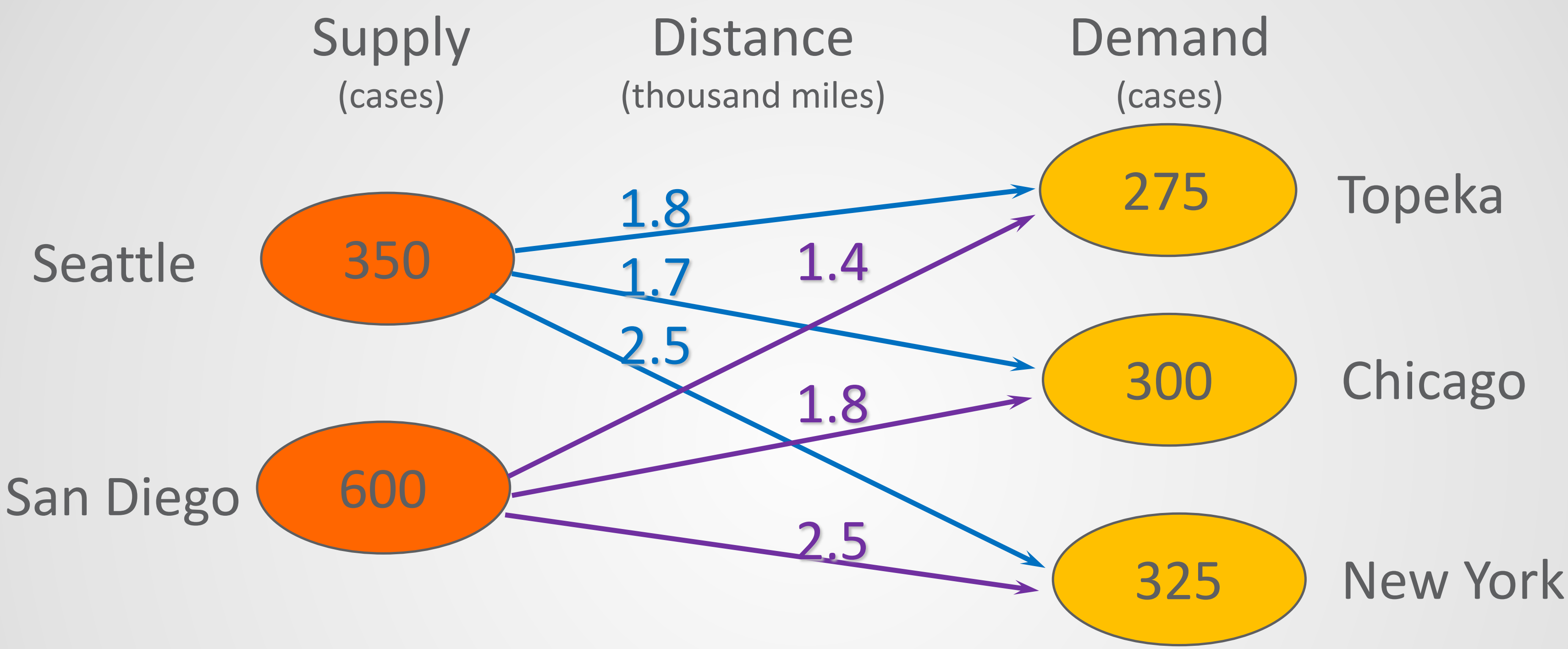
- GAMS at a Glance
- Foundation and Design Principle
- GAMS – A simple Example
- Wrap-Up

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

A Simple Transportation Problem



Freight: $90 case / thousand miles

Minimize	Transportation cost
subject to	Demand satisfaction at markets
	Supply constraints

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

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

subject to

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

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

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

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

GAMS Algebra (declarative Model)

```
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 ;
Scalar f  freight in dollars per case per thousand miles;

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 transport /all/ ;
```

Model is executable description of the problem

Model types in this example

LP

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

MIP

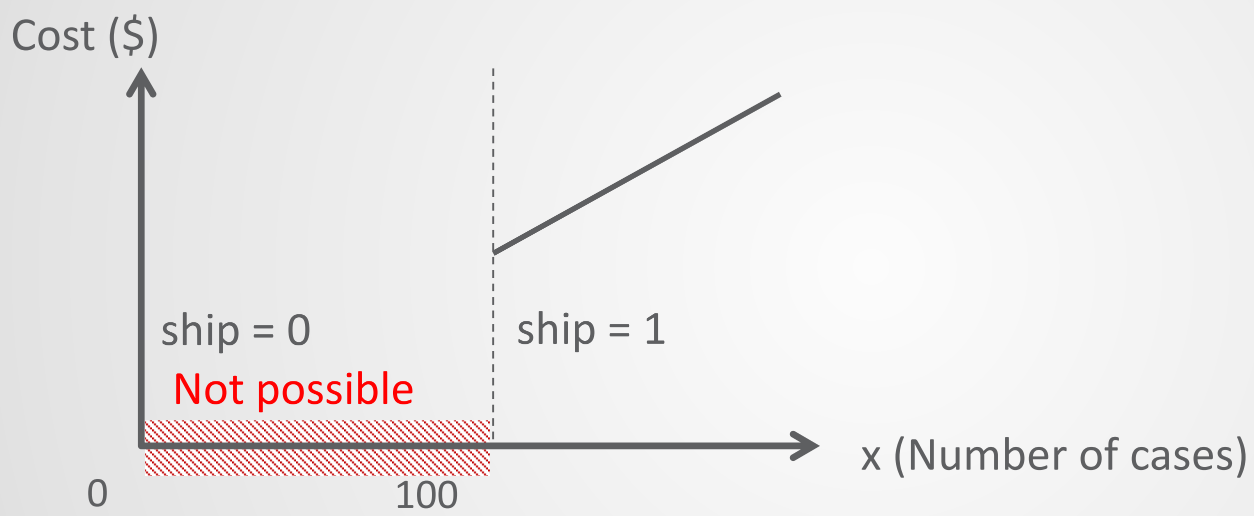
Allows discrete decisions,
e.g. if we ship, then we ship at least 100 cases.

MINLP

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

MIP Model: Minimum Shipment of 100 cases

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



add constraints:

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

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

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

MIP Model: GAMS Syntax

5_trnsport_LP_MIP_MINLP_SP.gms

```
* 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

5_trnsport_LP_MIP_MINLP_SP.gms | 5_trnsport_LP_MIP_MINLP_SP.lst | results.gdx

- Compilation
- Include File Summa
- Equation Listing S
- Equation
- Column Listing S
- Column
- Model Statistics SO
- Solution Report SO
- SolEQU
- SolVAR
- Equation Listing S
- Equation
- Column Listing S
- Column
- Model Statistics SO
- Solution Report SO
- SolEQU
- SolVAR
- Execution
- Display
  - rep

```
san-diego.topeka        .         1.000      1.000       EPS


**** REPORT SUMMARY :           0      NONOPT
                                0 INFEASIBLE
                                0  UNBOUNDED
GAMS 24.8.5  r61358 Released May 10, 2017 WEX-WEI x86 64bit/MS Windows 08/07
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n



----      66 PARAMETER rep  report parameter

                              LP          MIP

seattle  .new-york      50.000
seattle  .chicago      300.000      300.000
san-diego.new-york     275.000      325.000
san-diego.topeka       275.000      275.000



EXECUTION TIME       =        0.000 SECONDS


USER: Franz Nelissen
      GAMS Software GmbH
```

5_trnsport_LP_MIP_MINLP_SP.gms | 5_trnsport_LP_MIP_MINLP_SP.lst | results.gdx

Entry	Symbol	Type	Dim	Nr Elem
3	a	Par	1	2
4	b	Par	1	3
14	bigM	Par	0	1
6	c	Par	2	6
9	cost	Equ	0	1
5	d	Par	2	6
11	demand	Equ	1	3
1	i	Set	1	2
2	j	Set	1	3
17	maxship	Equ	2	6
13	minS	Par	0	1
16	minship	Equ	2	6
12	rep	Par	3	7

rep(i, j, *): report parameter

		LP	MIP
seattle	new-york	50	
	chicago	300	300
san-diego	new-york	275	325
	topeka	275	275

Model types **in this example**

LP

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

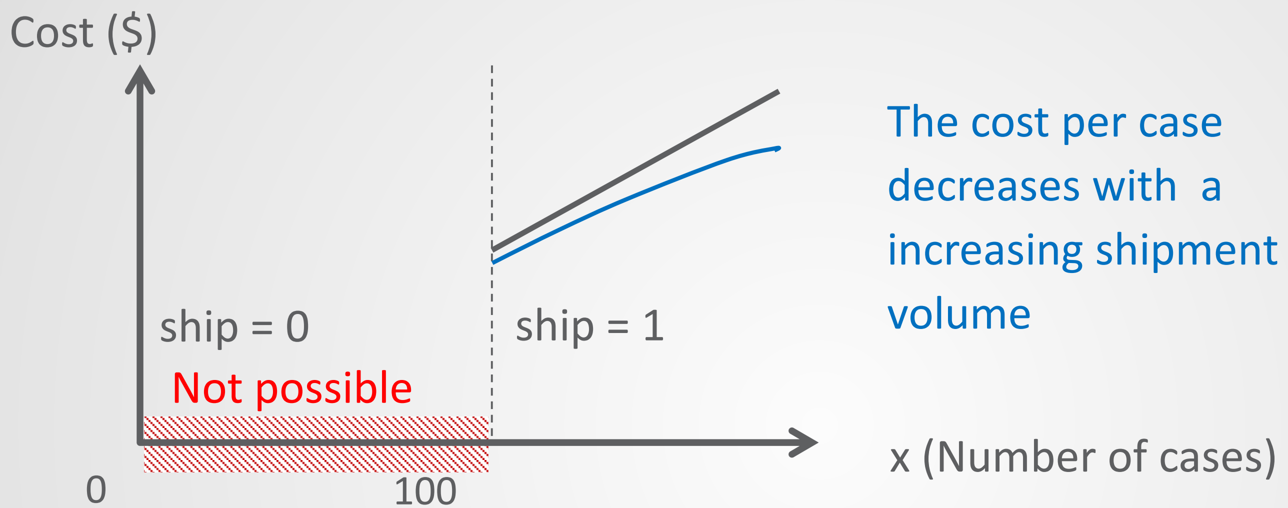
MIP

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

MINLP: Cost Savings



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)

MINLP Model: GAMS Syntax

5_trnsport_LP_MIP_MINLP_SP.gms | 5_trnsport_LP_MIP_MINLP_SP.lst | results.gdx

```
* 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

5_trnsport_LP_MIP_MINLP_SP.gms | 5_trnsport_LP_MIP_MINLP_SP.lst | results.gdx

- Compilation
- Include File Summar
- Equation Listing SO
- Equation
- Column Listing SO
- Column
- Model Statistics SO
- Solution Report SO
- SolEQU
- SolVAR
- Equation Listing SO
- Equation
- Column Listing SO
- Column
- Model Statistics SO
- Solution Report SO
- SolEQU
- SolVAR
- Execution
- Display
- Equation Listing SO
- Equation
- Column Listing SO
- Column

```
                              0 INFEASIBLE
                              0  UNBOUNDED
                              0     ERRORS
GAMS 24.8.5  r61358 Released May 10, 2017 WEX-WEI x86 64bit/MS Windows 08/07
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----      85 PARAMETER rep  report parameter

                            LP         MIP       MINLP

seattle  .new-york      50.000
seattle  .chicago      300.000     300.000     300.000
san-diego.new-york     275.000     325.000     325.000
san-diego.topeka       275.000     275.000     275.000
```

```
                           LOWER     LEVEL     UPPER

---- VAR z                 -INF     153.675     +INF
```

```
                           LOWER     LEVEL     UPPER

---- VAR z                 -INF     115.438     +INF
```

5_trnsport_LP_MIP_MINLP_SP.gms | 5_trnsport_LP_MIP_MINLP_SP.lst | results.gdx

Entry	Symbol	Type	Dim	Nr Elem
3	a	Par	1	2
4	b	Par	1	3
18	beta	Par	0	1
14	bigM	Par	0	1
6	c	Par	2	6
9	cost	Equ	0	1
19	costnlp	Equ	0	1
5	d	Par	2	6

rep(i, j, *): report parameter

Plane Inde

		LP	MIP	MINLP
seattle	new-york	50		
	chicago	300	300	300
san-diego	new-york	275	325	325
	topeka	275	275	275

Agenda

- GAMS at a Glance
- Foundation and Design Principle
- GAMS – A simple Example
- Wrap-Up

Thank You

Meet us at the GAMS booth!