GAMS – An Introduction

Get ready to learn the basics of GAMS

Frederik Fiand & Lutz Westermann
GAMS Software GmbH

Agenda

GAMS at a Glance

GAMS - Hands On Examples

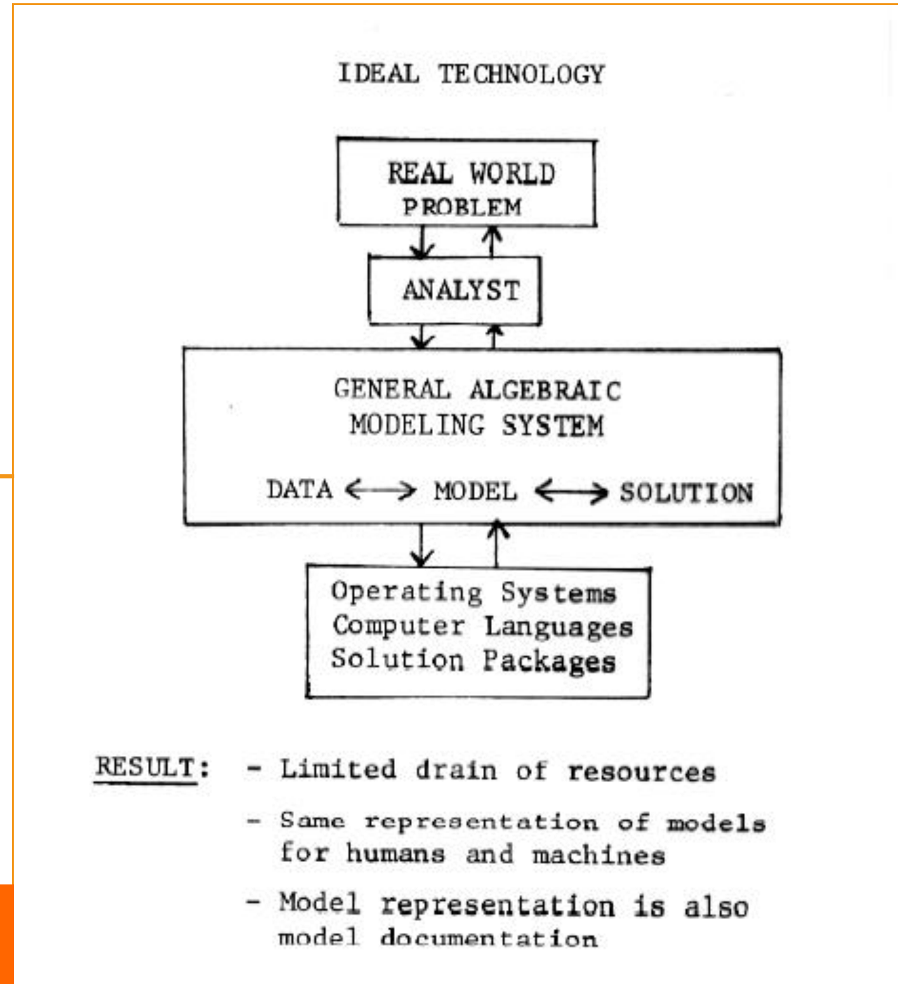
Outlook 1 - Deployment of GAMS models

Outlook 2 - Solving Multiple Models Efficiently

GAMS at a Glance

1976 - A World Bank Slides

The Vision



GAMS came into being!

The aim of this system is to provide one representation of a model which is easily understood by both humans and machines.

[J. Bischopp, A. Meeraus, On the Development of a General Algebraic Modeling System in a Strategic Planning Environment. *Mathematical Programming Study* 20 (***1982***) 1–29.]

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

[M. Bussieck & A. Meeraus, Algebraic Modeling for IP and MIP (GAMS), Annals of Operations Research 149(1): *History of Integer Programming: Distinguished Personal Notes and Reminiscences, Guest Editors: Kurt Spielberg and Monique Guignard-Spielberg*, February, **2007**, pp. 49-56]

Company

- Roots: World Bank, 1976
- Went commercial in 1987
- Locations
  - GAMS Development Corporation (Washington)
  - GAMS Software GmbH (Germany)
- Product
  - The **G**eneral **A**lgebraic **M**odeling **S**ystem

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

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

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

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

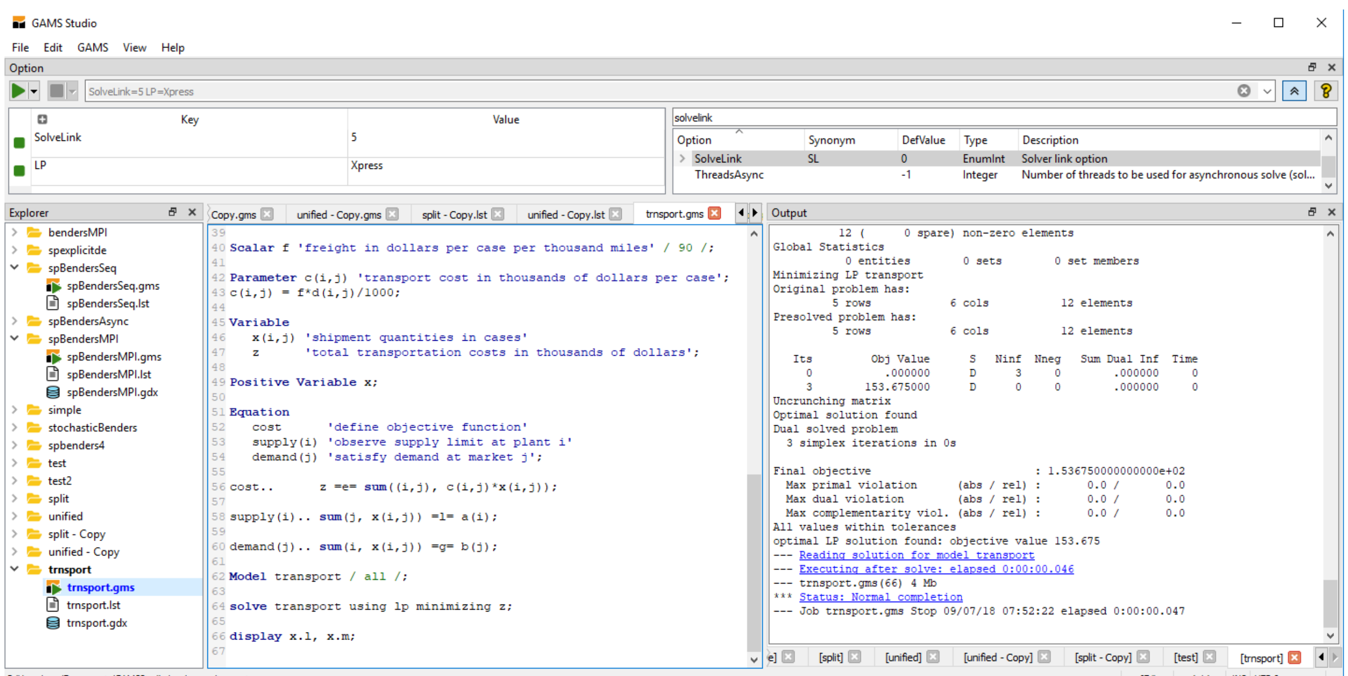
Procedural elements

- Control Flow Statements (e.g. loops, for, if,...),
- Build complex problem algorithms within GAMS
- Simplified interaction with other systems
  - Data exchange
  - APIs

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 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

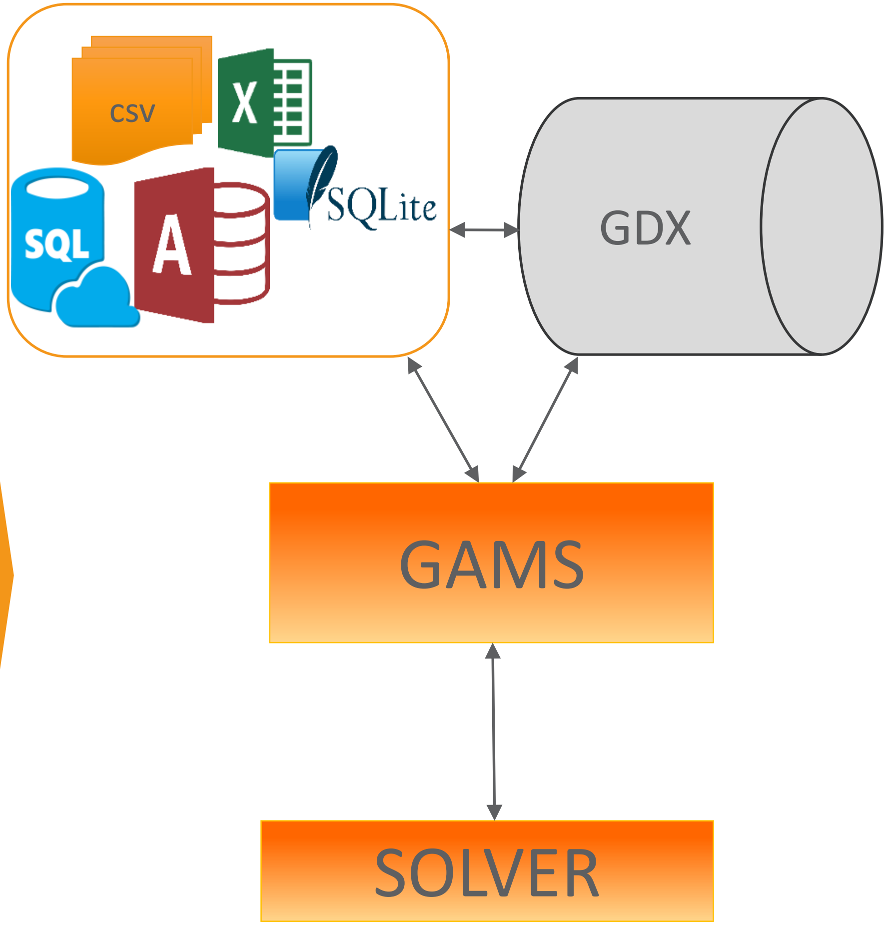
FICO · Gurobi Optimization · IBM · mosek · ... · COIN|OR

➡ Switching between solvers with one line of code!

Independence of Model and Data

- Declarative Modeling
- ASCII: Initial model development

- GDX: Data layer ("contract") between GAMS and applications
  - Platform independent
  - No license required
  - Direct GDX interfaces and general API
  - …

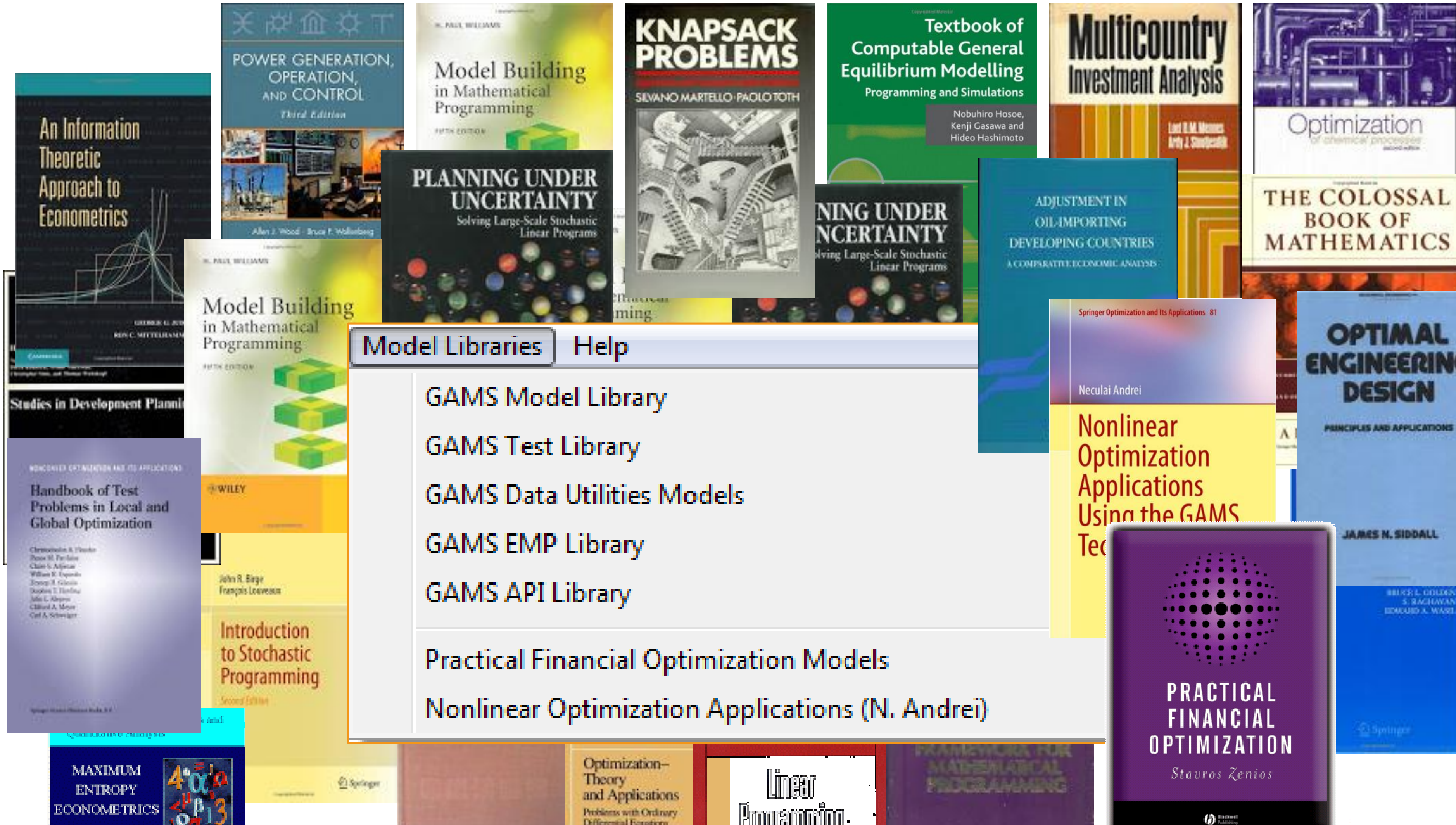


Independence of Model and User Interface

API's

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

Free Model Libraries



➤ **More than 1400 models!**

Why GAMS?

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

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

- Open for new developments
- Protecting investments of users

GAMS –
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

SP
- Uncertainty
- E.g.: Uncertain demand

A Simple Transportation Problem

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

Seattle (350)

San Diego (600)

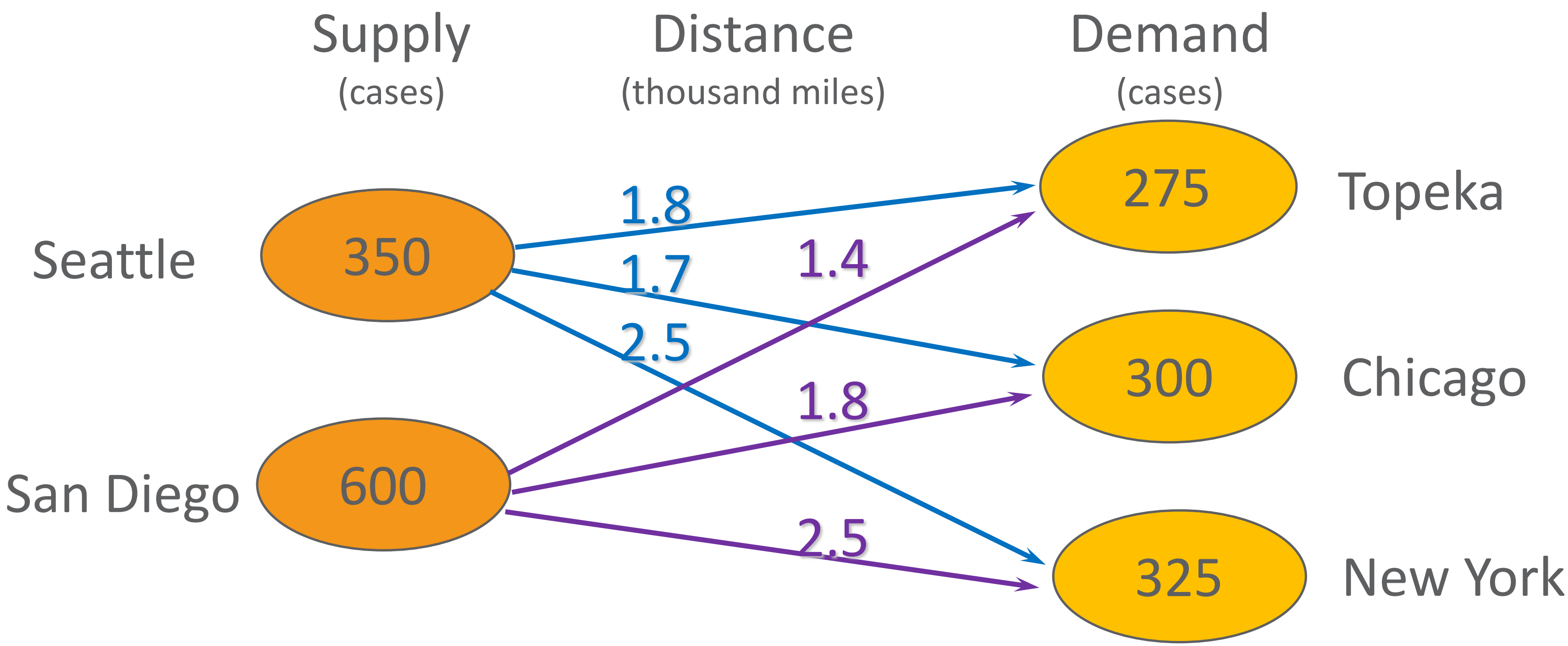
Topeka (275)

Chicago (300)

New York (325)

Freight: $90 case / thousand miles

A Simple Transportation Problem



Freight: $90 case / thousand miles

Minimize	Transportation cost
subject to	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_i	(Demand in cases)

$\min \quad \sum_i \sum_j c_{ij} \cdot x_{ij}$ (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 Syntax (LP Model)

GAMS Studio

File Edit GAMS View Help

Explorer

5_trnsport_LP_MIP_MINLP_SP

5_trnsport_LP_MIP_MINLP_SP.gms

blend

trnsport

dummy

d2

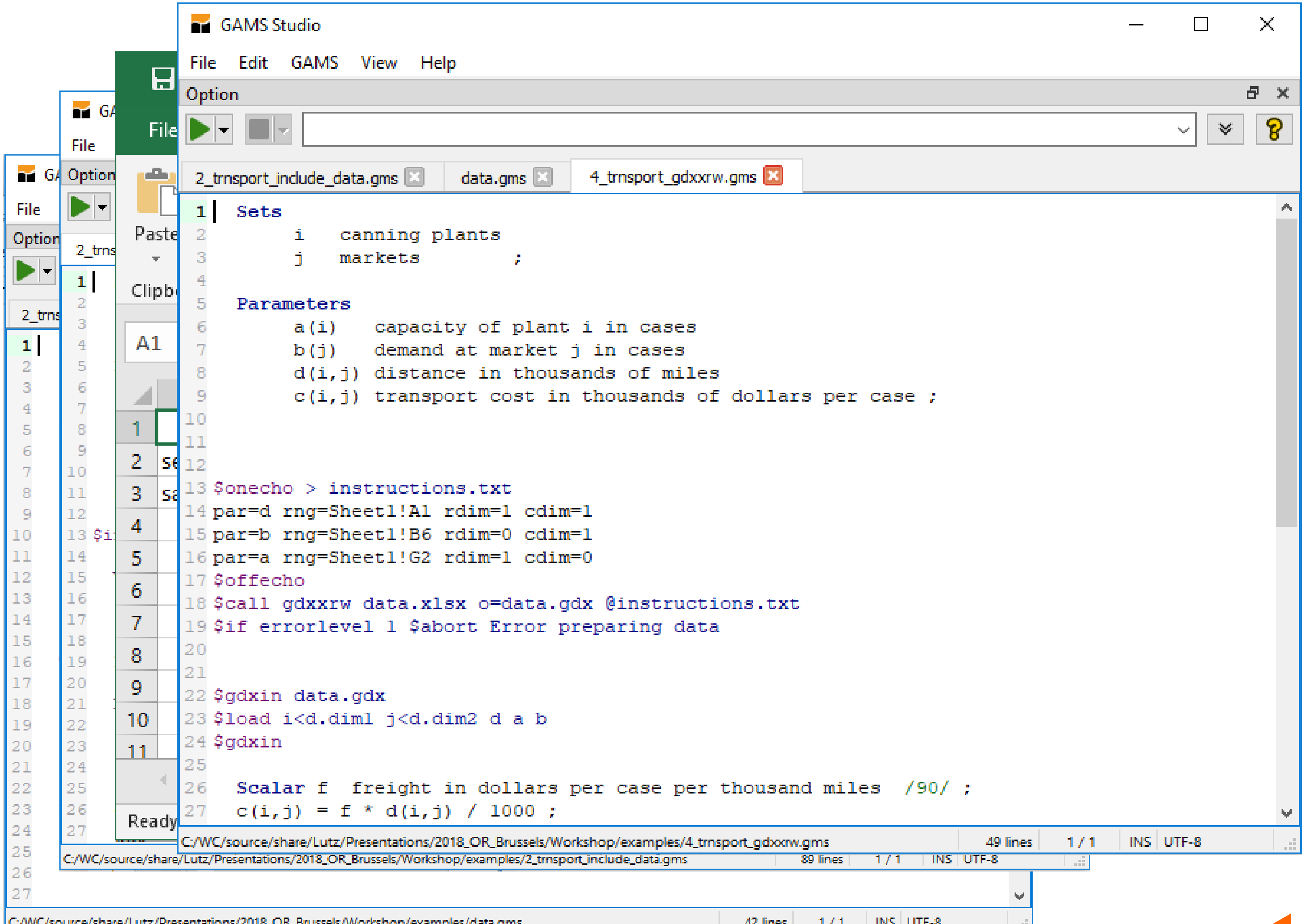
Option

5_trnsport_LP_MIP_MINLP_SP.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 transportLP /all/ ;

Solve transportLP using lp minimizing z ;
```

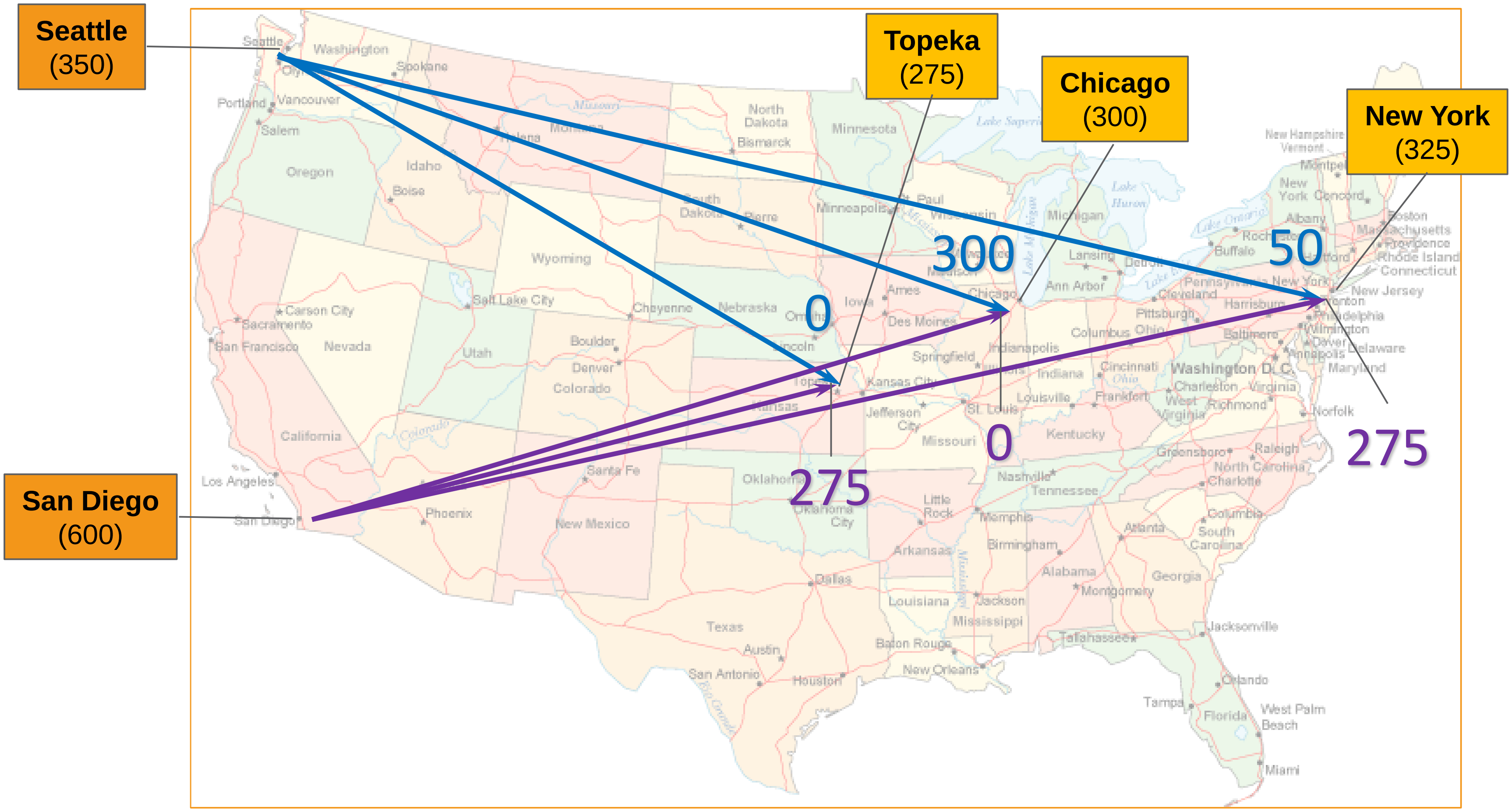
Hands-On

GAMS Syntax (Data Input)



Solution to LP model

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



Freight: $90 case / thousand miles

Total cost: $153,675

LP

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

MIP

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

MINLP

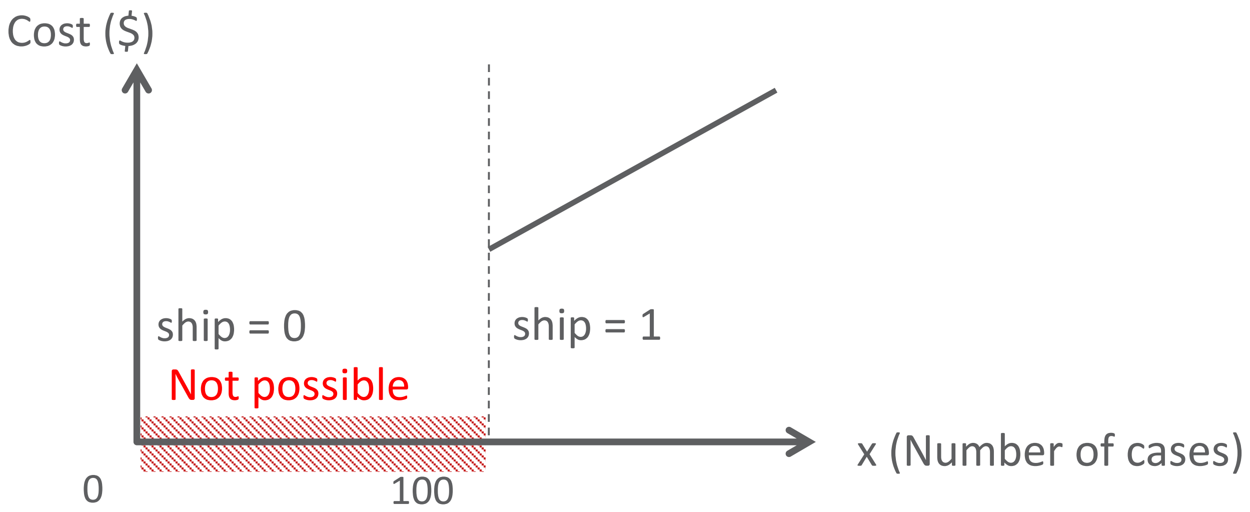
- Non-linearity
- E.g.: Decrease in unit cost with growing volumes

SP

- Uncertainty
- E.g.: Uncertain demand

MIP Model:
Minimum Shipment of 100 cases

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



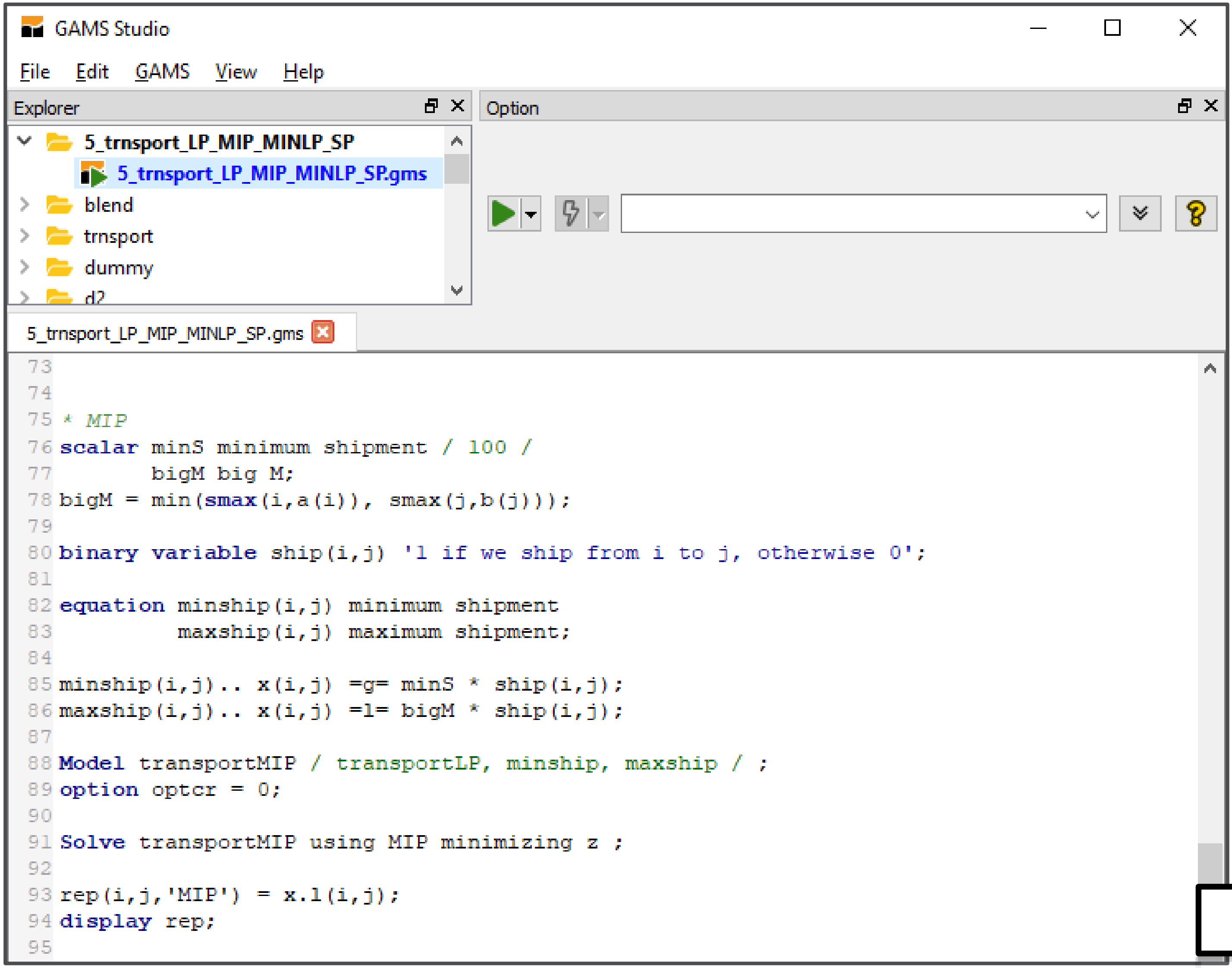
add constraints:

$x_{i,j} \geq 100 \cdot ship_{i,j} \quad \forall\, i,j$ (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)

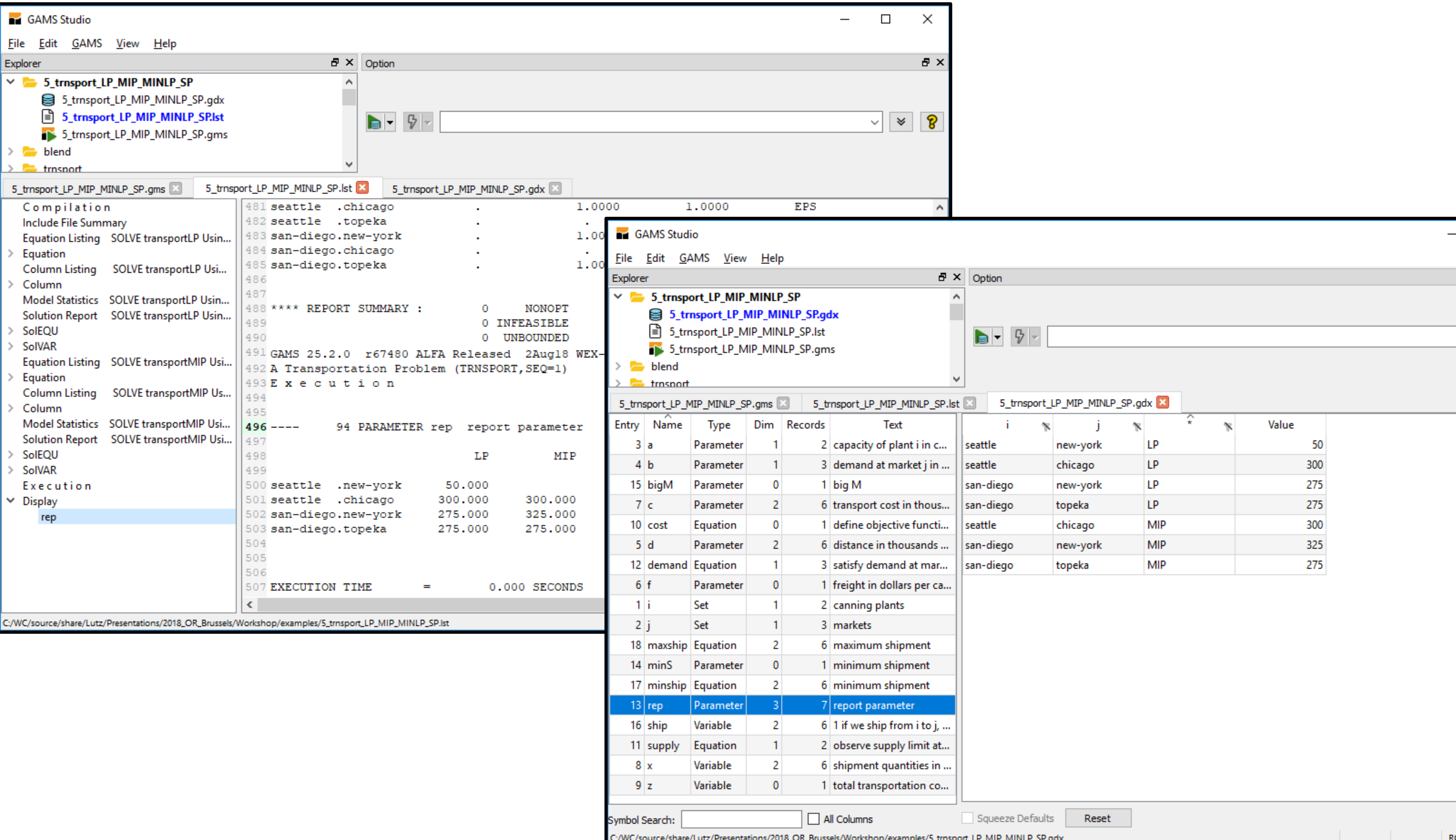
$ship_{i,j} \in \{0,1\}$

MIP Model: GAMS Syntax



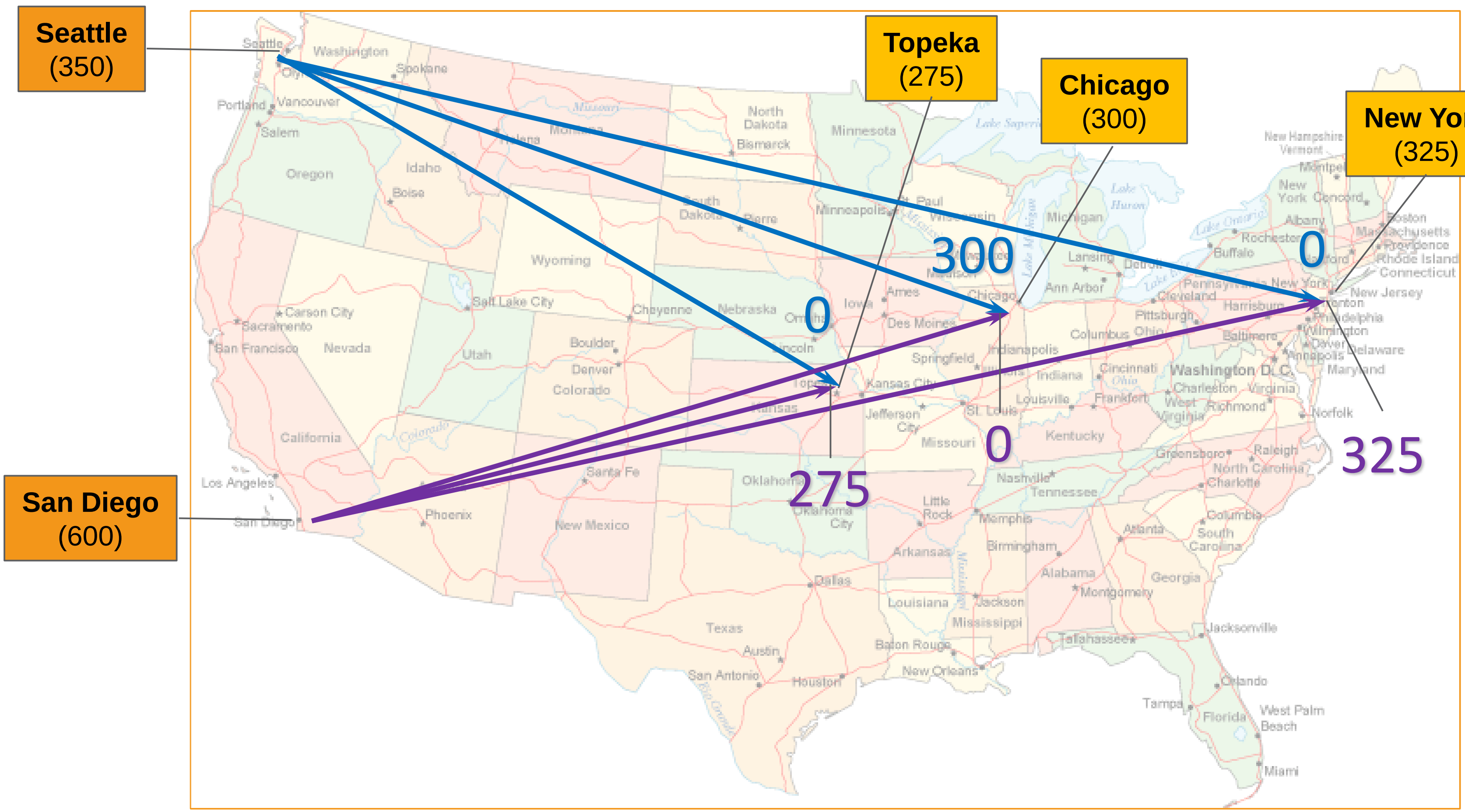
Hands-On

MIP Model: Results



MIP Model: Solution

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



Freight: $90 case / thousand miles

Total cost: $153,675

LP

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

MIP

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

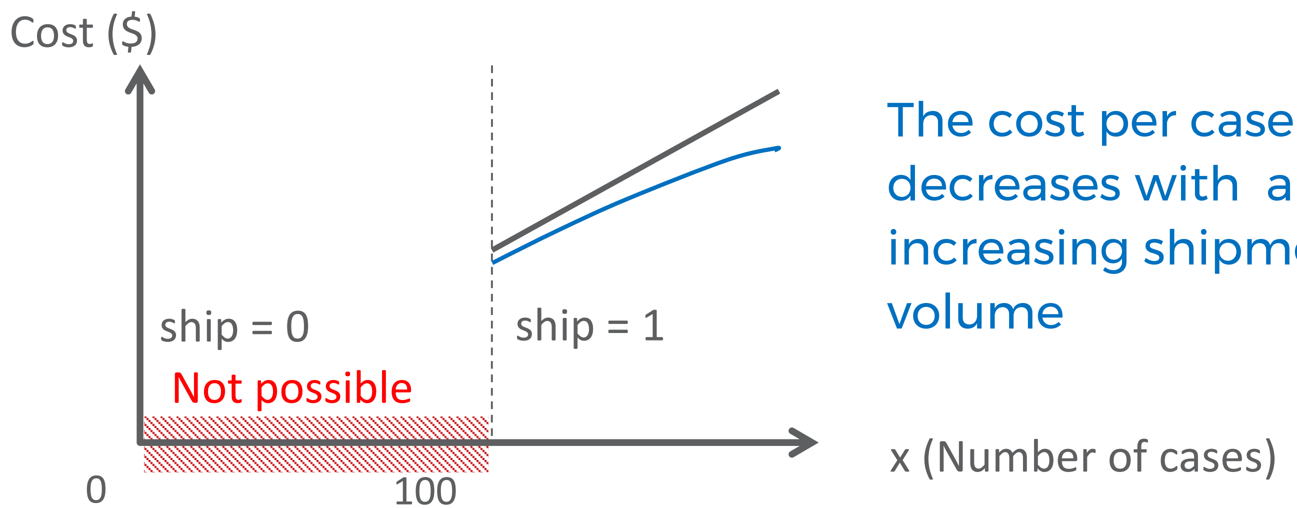
MINLP

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

SP

- Uncertainty
- E.g.: Uncertain demand

MINLP: Cost Savings



The cost per case decreases with a increasing shipment volume

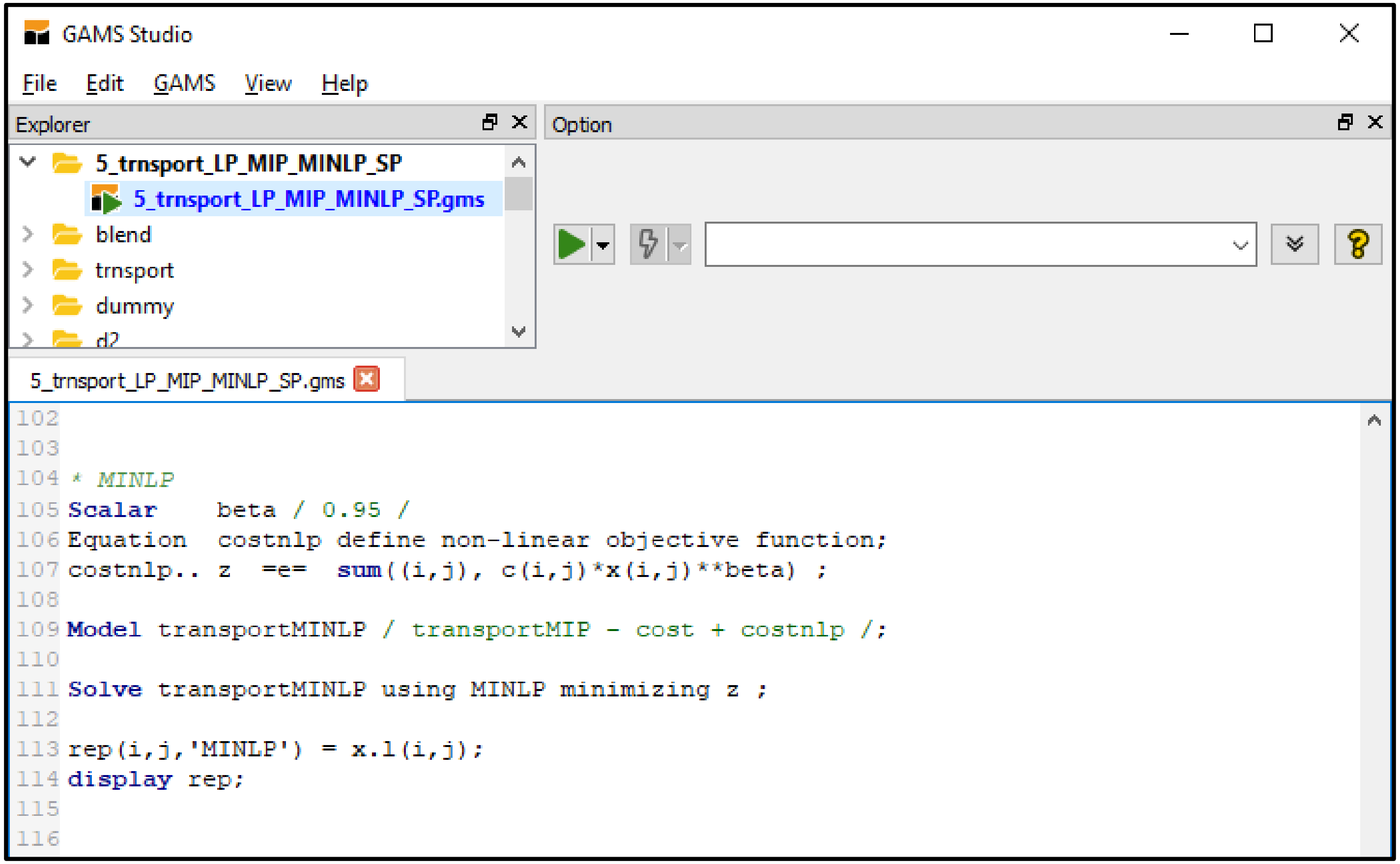
Replace:

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

With

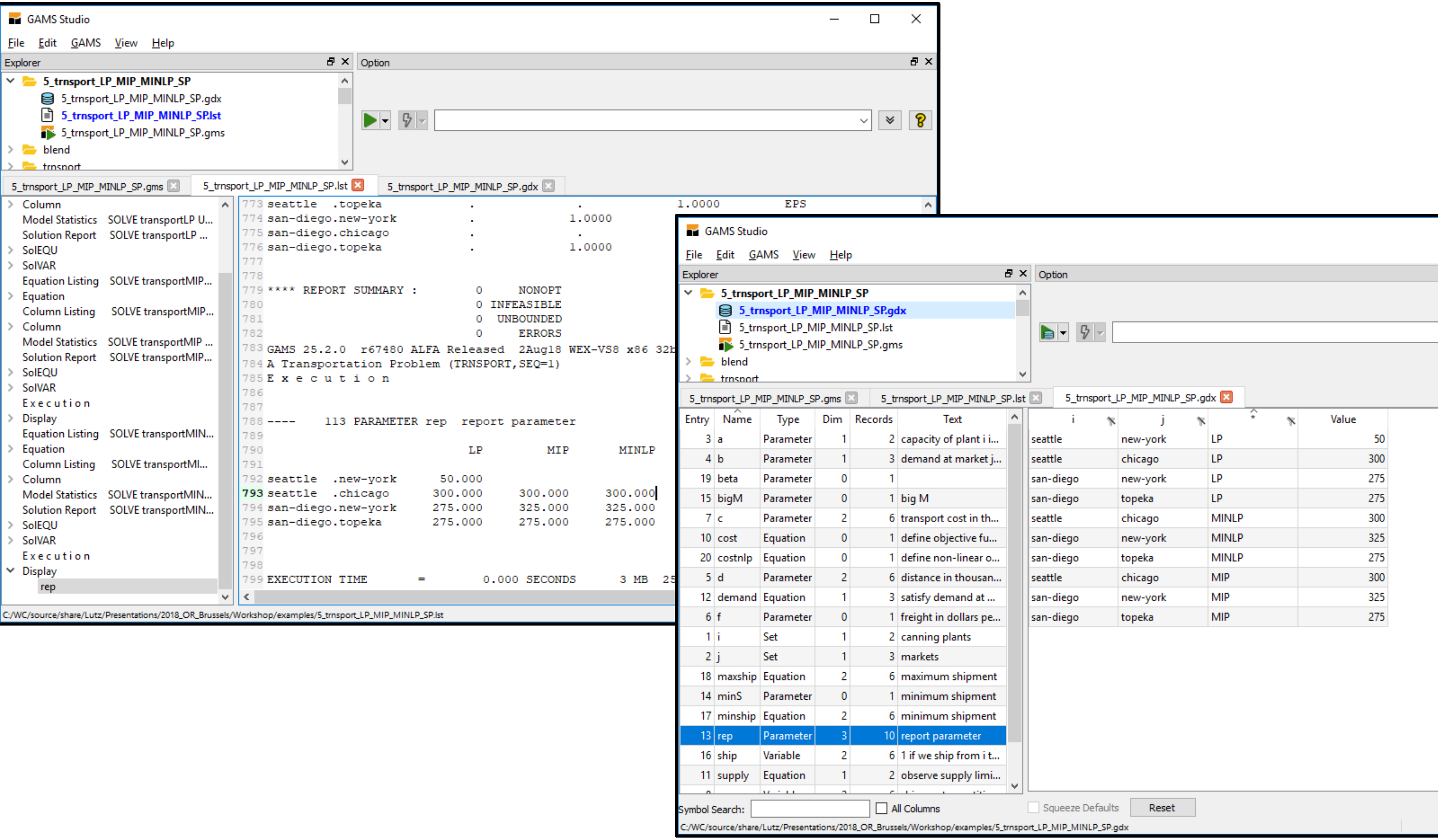
min $\sum_i \sum_j c_{ij} \cdot {x_{ij}}^{\boldsymbol{beta}}$ (Minimize total transportation cost)

MINLP Model: GAMS Syntax



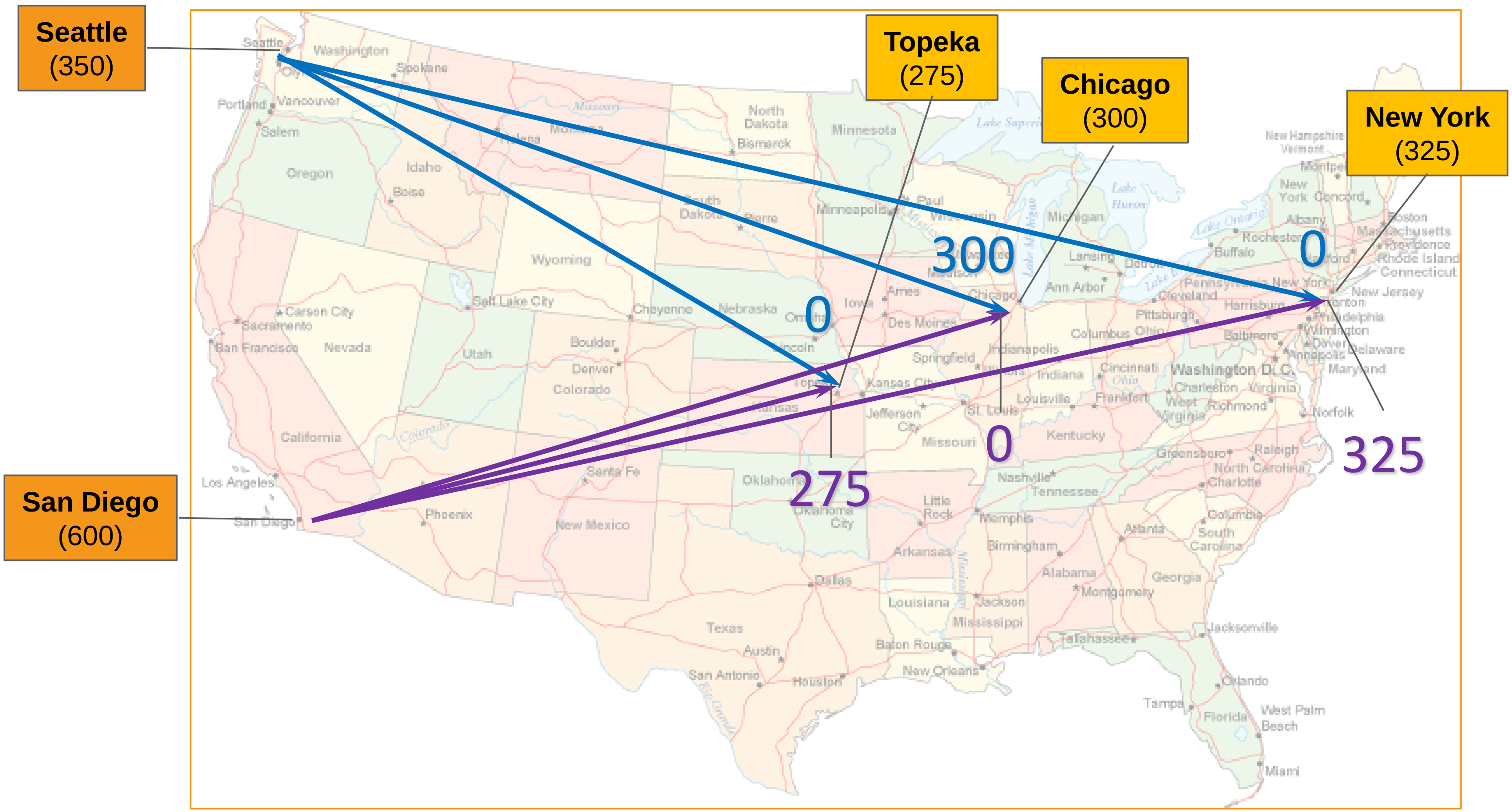
Hands-On

MINLP Model: Results



MINLP Model: Solution

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



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

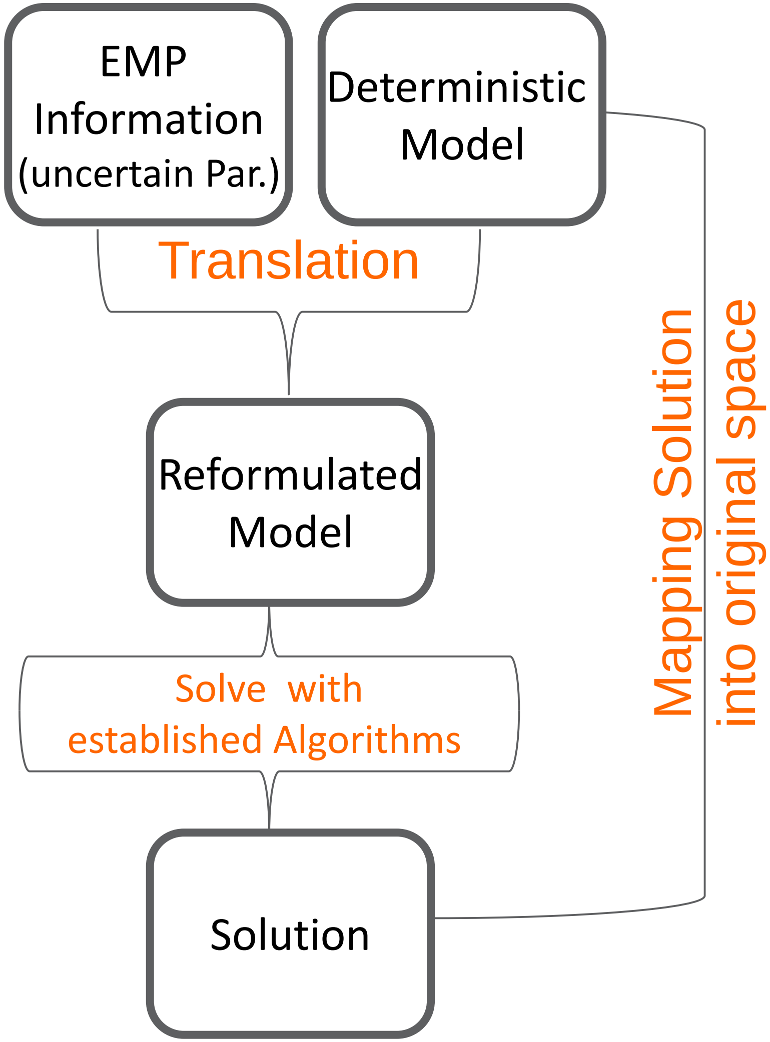
SP

- Uncertainty
- E.g.: Uncertain demand

Stochastic Programming in GAMS

EMP/SP

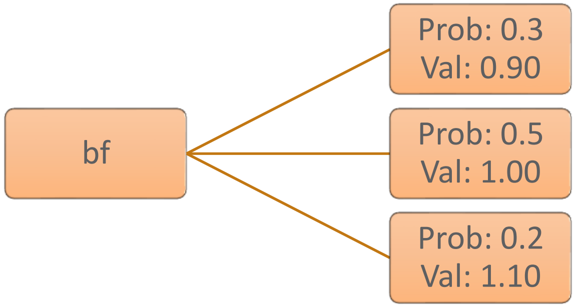
- Simple interface to add uncertainty to existing deterministic models
- (EMP) Keywords to describe uncertainty include: discrete and parametric random variables, stages, chance constraints, Value at Risk, …
- Available solution methods:
  - Automatic generation of **D**eterministic **E**quivalent (can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



Transport Example – Uncertain Demand

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

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:

```
costsp ..  z  =e=  sum((i,j), c(i,j)*x(i,j))+ sum(j,0.3*u(j));
demandsp(j) .. sum(i, x(i,j))  =g=  bf*b(j) - u(j) ;
```

Uncertain Demand: GAMS Algebra

GAMS Studio

File Edit GAMS View Help

Explorer

5_trnsport_LP_MIP_MINLP_SP

5_trnsport_LP_MIP_MINLP_SP.gms

blend

trnsport

dummy

d2

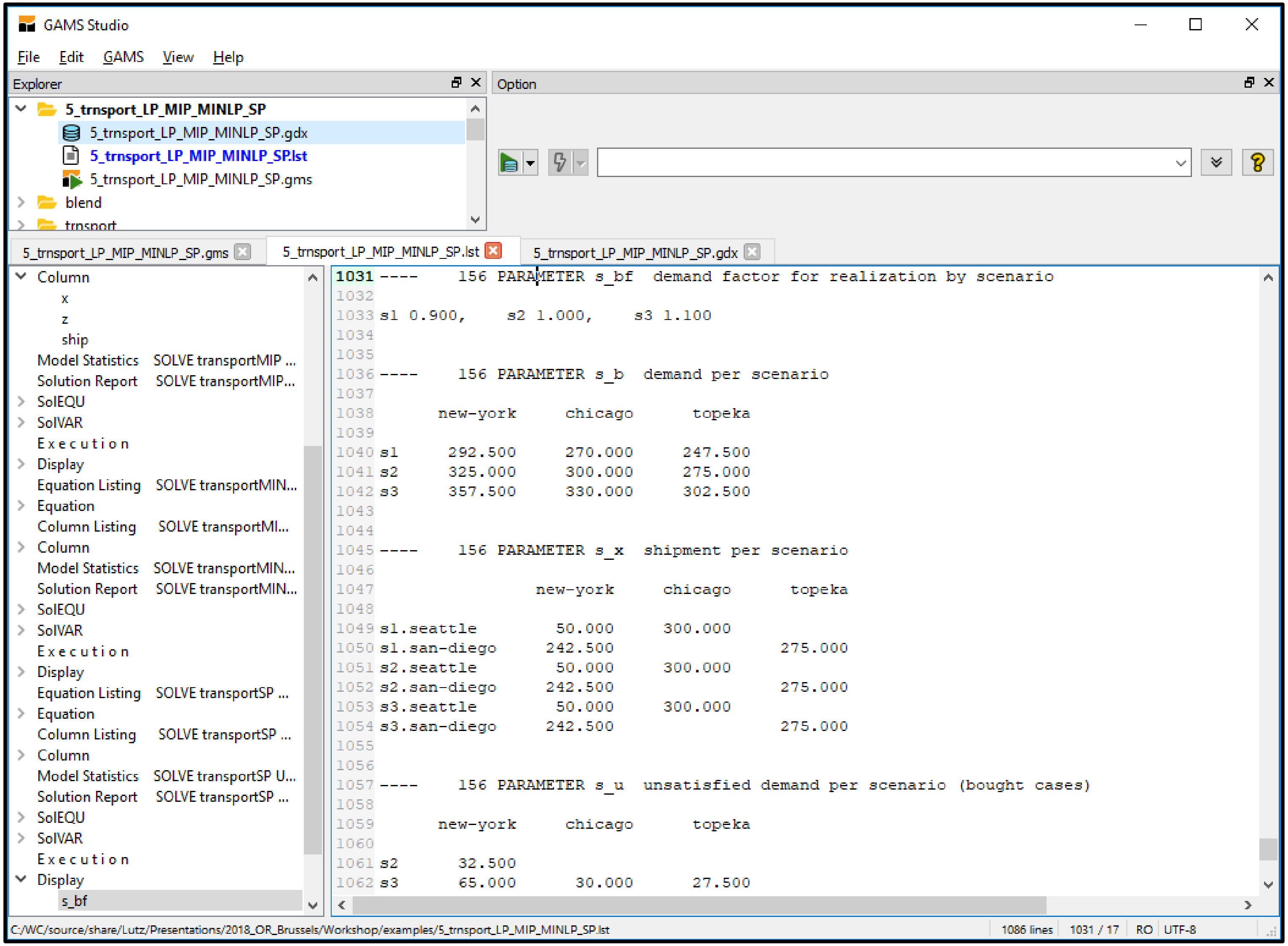
Option

5_trnsport_LP_MIP_MINLP_SP.gms

```
* Stochastic Program with uncertain demand
Positive variable u(j) unsatisfied demand;
Scalar   bf demand factor / 1 /;
Equation costsp      define objective function for SP
         demandsp(j) demand satisfaction in SP;

costsp..       z =e= sum((i,j), c(i,j)*x(i,j)) + sum(j, 0.3*u(j));
demandsp(j)..  sum(i, x(i,j)) =g= bf*b(j) - u(j);

Model transportSP / costsp, demandsp, supply /;
File emp / '%emp.info%' /; put emp;
$onput
randvar bf discrete 0.3  0.9
                    0.5  1.0
                    0.2  1.1
stage 2 bf u demandsp
$offput
Putclose emp;

Set scen scenarios / s1*s4 /;
Parameter
    s_bf(scen)     demand factor for realization by scenario
    s_x(scen,i,j)  shipment per scenario
    s_u(scen,j)    unsatisfied demand per scenario (bought cases);

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

option emp=lindo;
Solve transportSP min z use emp scenario dict;
```

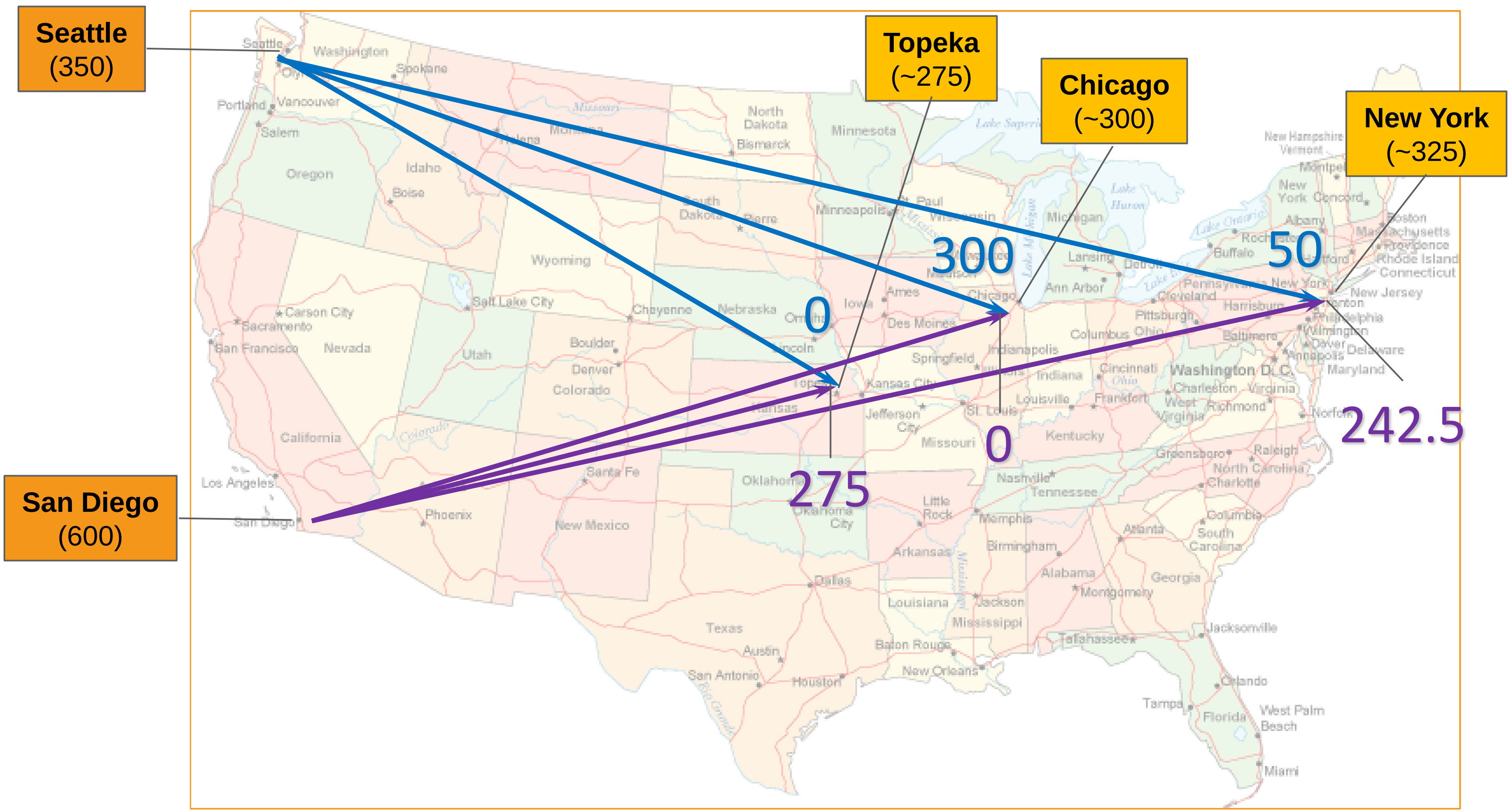
Hands-On

Uncertain Demand: Results



Stochastic Program: Solution

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



Freight: $90 case / thousand miles

Total cost: $158,588

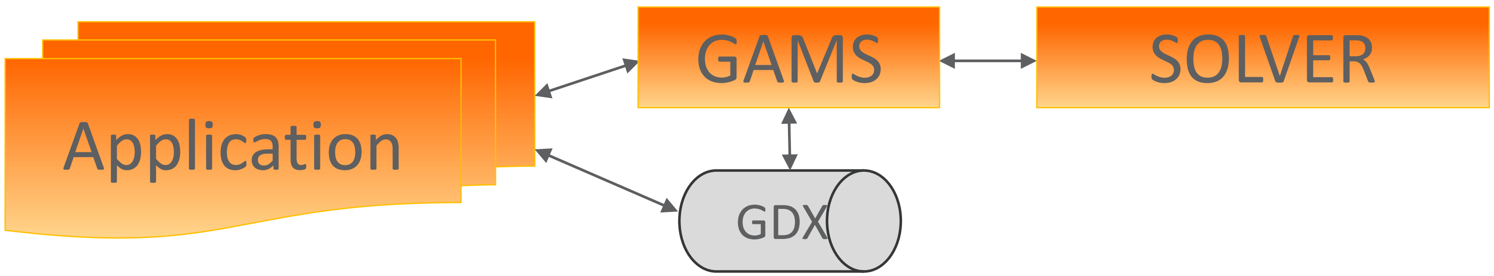
Stochastic Programming in GAMS

- 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: https://www.gams.com/latest/docs/UG_EMP_SP.html

Outlook: Deployment of GAMS models

- APIs – Application Programming Interfaces to GAMS
- Using R/Shiny to deploy and visualize GAMS models in a Web Interface
- Using GAMS Jupyter Notebooks to tell “optimization stories”

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

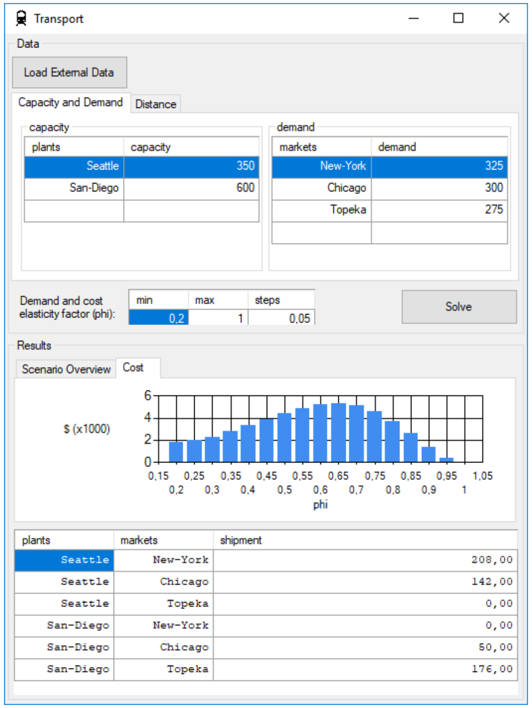
- Low level APIs
  - GDX, OPT, GAMSX, GMO, …
  - High performance and flexibility
  - Automatically generated imperative APIs for several languages (C, C++, C#, Delphi, Java, Python, VBA, …)

- 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

Transport Application GUI Example

- Scenario solves of the transportation problem

- Features:
  - Preparation of input data
  - Loading data from Access file
  - Solving multiple scenarios of a model
  - Displaying results



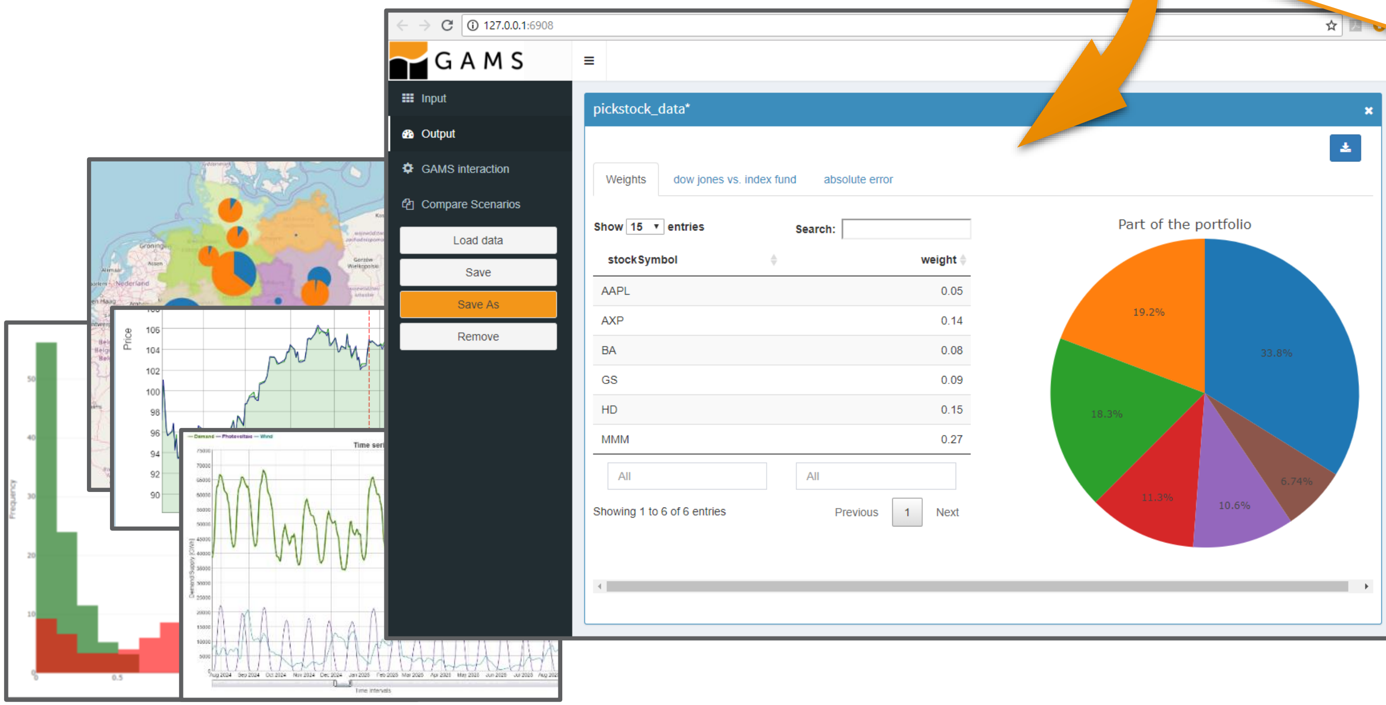
- Four implementation steps:
  1. Graphical User Interface
  2. Preparation of GAMS model
  3. Implementation of scenario solving using GAMSJob
  4. GAMSModelInstance for performance improvements

Hands-On

From GAMS Model to Visual Web User Interface

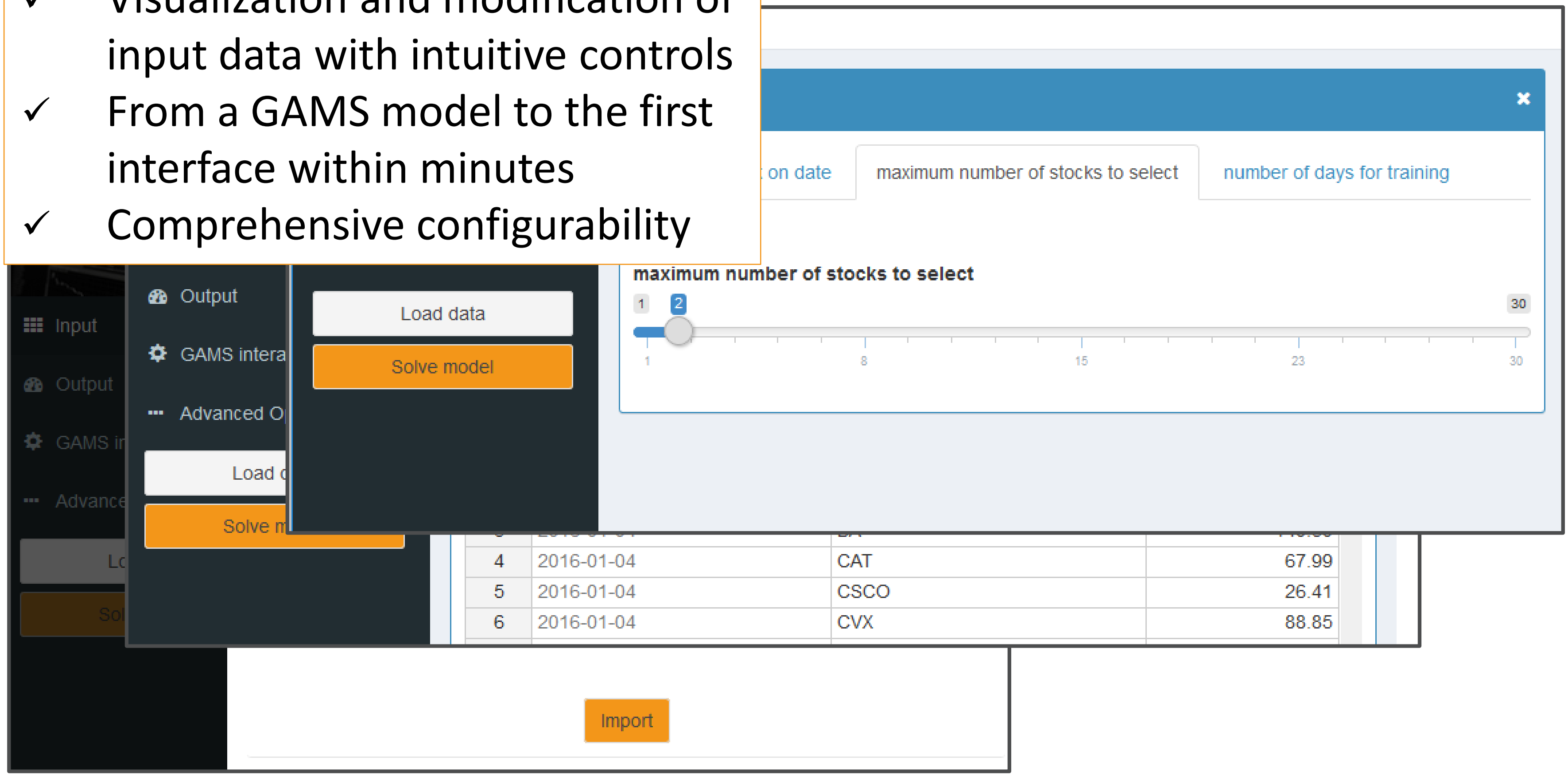
```
Equation
    deffit(date)     'fit to Dow Jones index'
    defpick(symbol) 'can only use stock if picked'
    defnumstock      'few stocks allowed'
    defobj           'absolute violation (L1 norm) from index';

deffit(ds)..  sum(s, price(ds,s)*w(s)) =e= index(ds) + slpos(ds) - slneg(ds);

defpick(s)..  w(s) =l= p(s);

defnumstock.. sum(s, p(s)) =l= maxstock;

defobj..      obj =e= sum(ds, slpos(ds) + slneg(ds));
```

Currently under Development

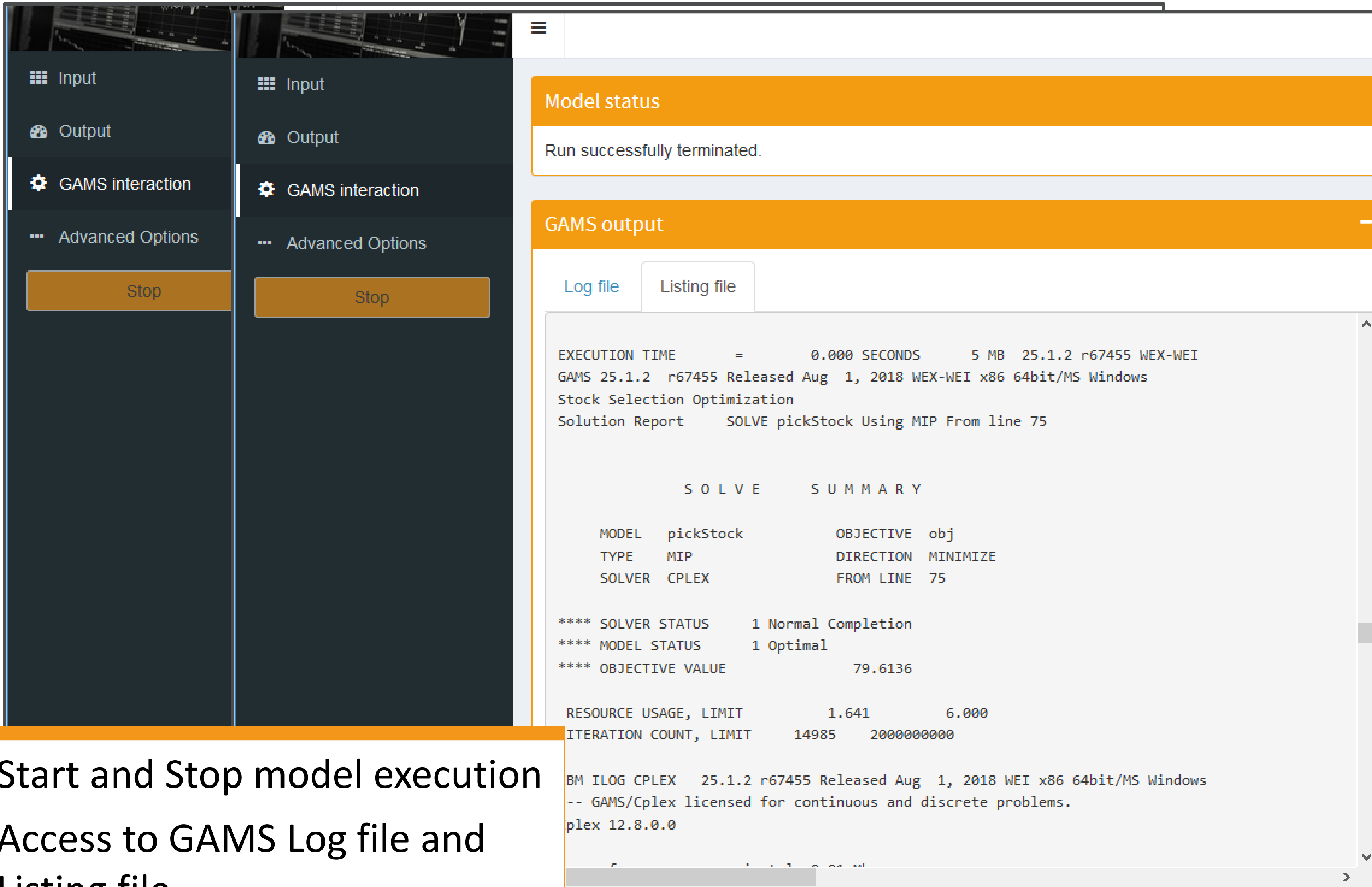


Setting the Model Input Data

- ✓ Data exchange via local files or database connection
- ✓ Visualization and modification of input data with intuitive controls
- ✓ From a GAMS model to the first interface within minutes
- ✓ Comprehensive configurability

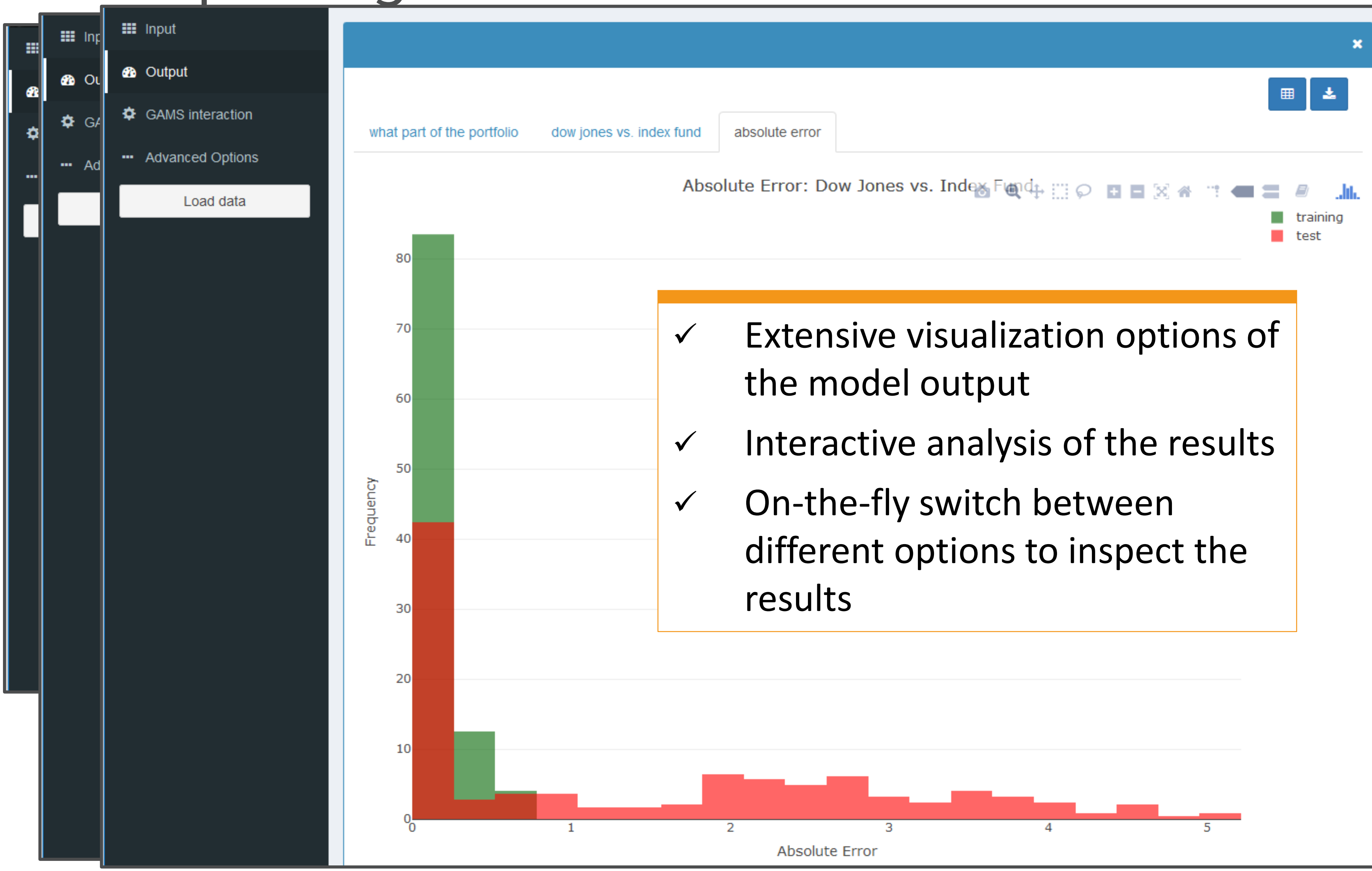


Communication with GAMS



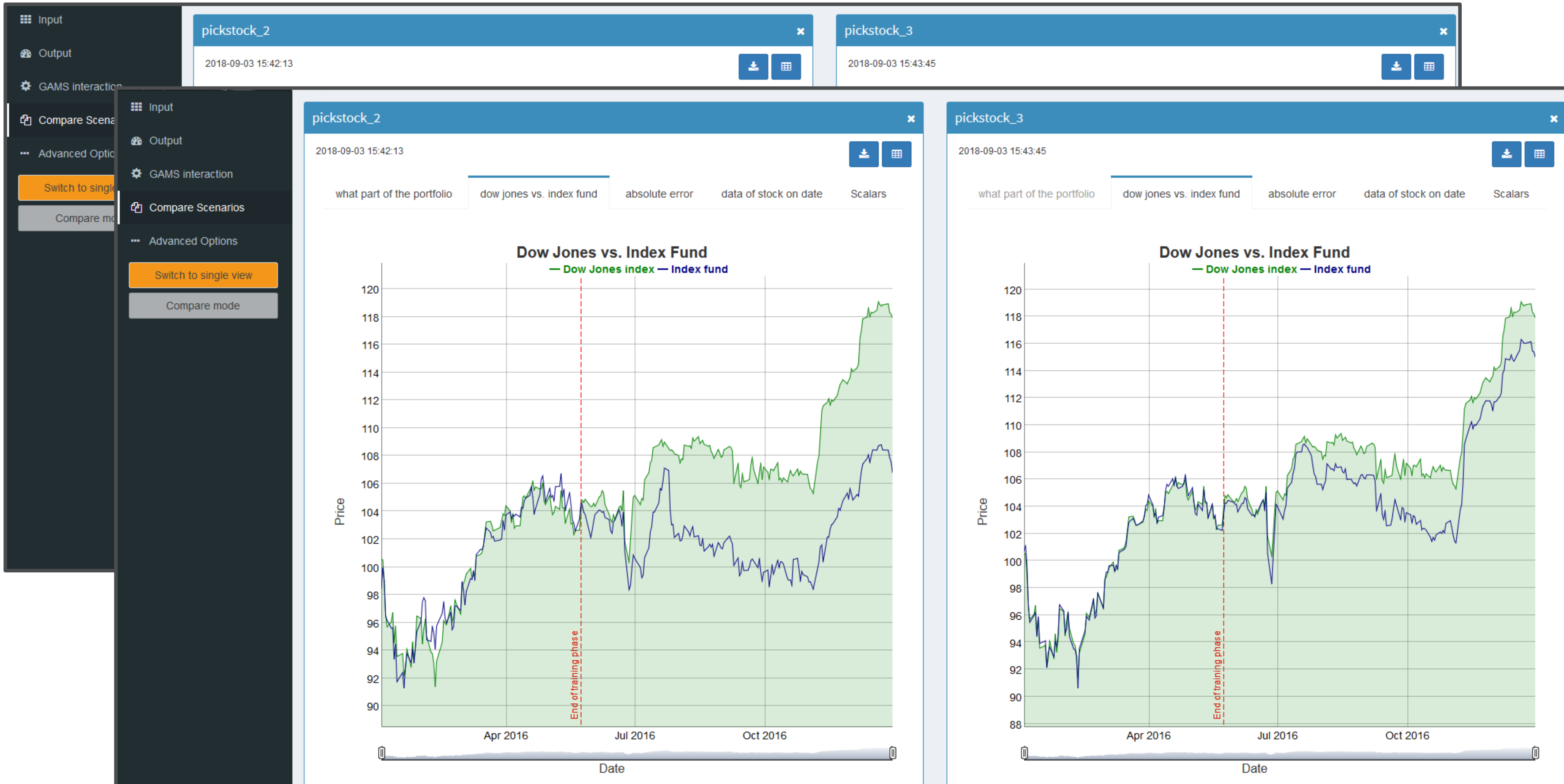
- ✓ Start and Stop model execution
- ✓ Access to GAMS Log file and Listing file

Inspecting the Results



- ✓ Extensive visualization options of the model output
- ✓ Interactive analysis of the results
- ✓ On-the-fly switch between different options to inspect the results

Scenario Management



✓ Solve multiple scenarios or load saved data for comparison

Hands-On

Enhanced Model Deployment in GAMS using R/Shiny

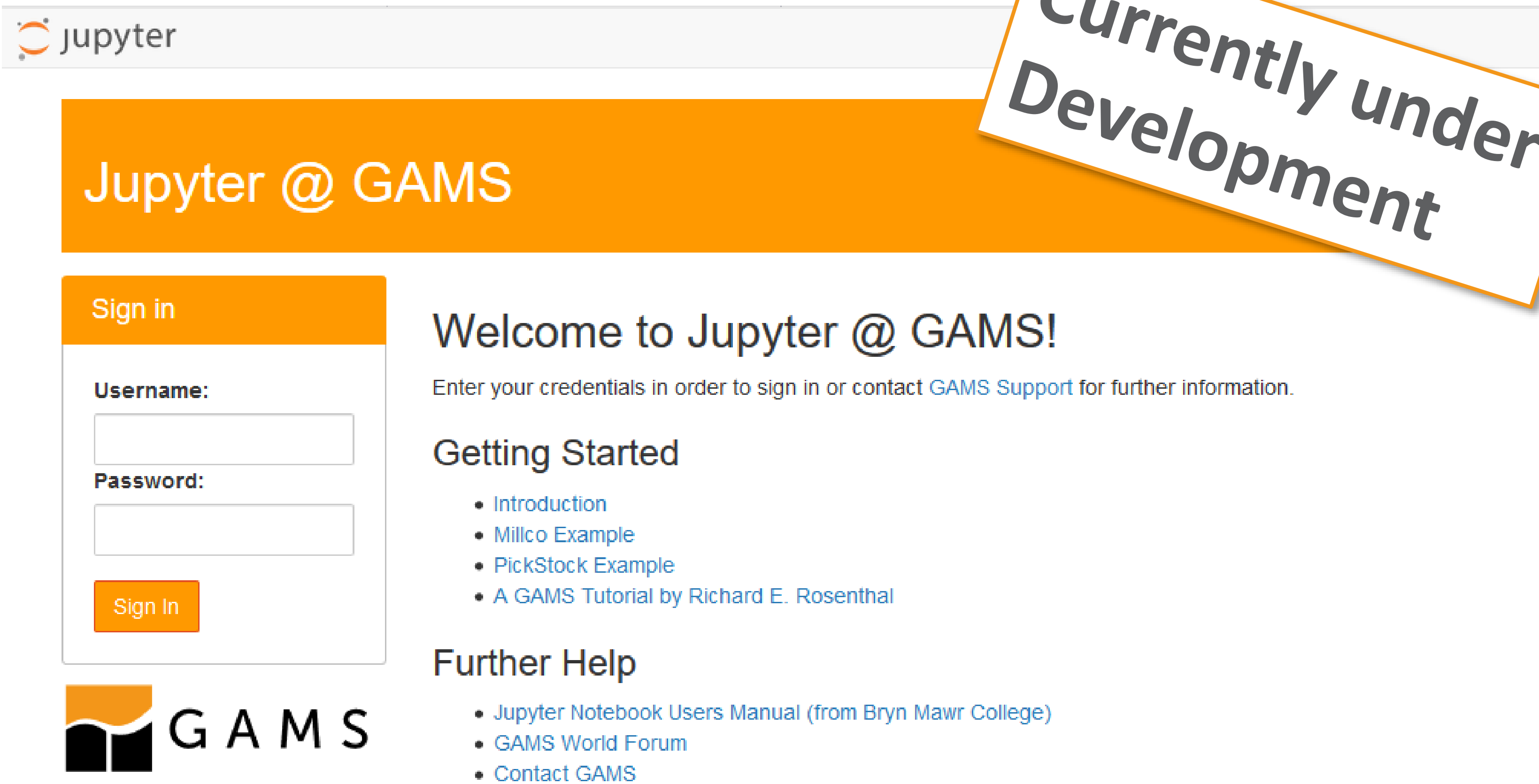
Model Deployment in GAMS
Wednesday, September 12, 4:10 pm - 5:50 pm
(Session WD-16)

- Application connects W… …ce with a GAMS model
- User Interface allows
  - ✓ Data exchange via local files or data…
  - ✓ Modification of the input data
  - ✓ Extensive visualization options
  - ✓ Comparison of different scenarios
  - ✓ Multi-user support based on Docker technology
  - ✓ User authentication
- Tool with intuitive interface for planners (configuration vs. programming)

- This “product” is currently under development. If you are interested in getting involved, please contact support@gams.com

Welcome to Jupyter @ GAMS!

Currently under Development



GAMS Jupyter Example

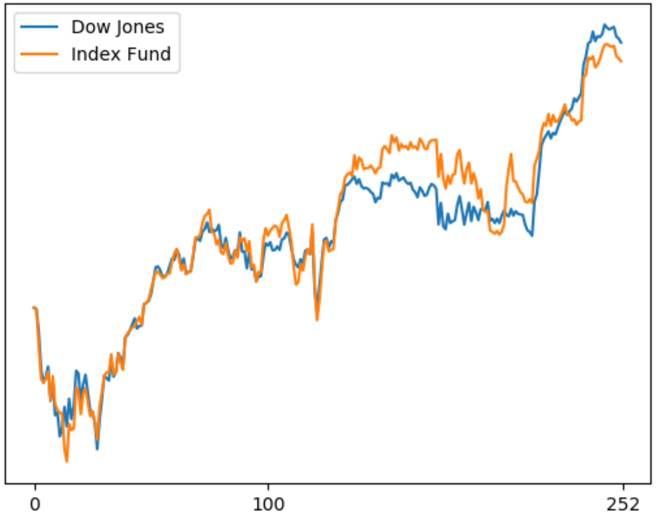
jupyter nbviewer

JUPYTER FAQ

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

Plotting of the results

In [18]:
```
%gams_pull -d fund error
fig, ax = plt.subplots()
index.plot(y="value", ax=ax, xticks=[0, trainingDays, len(date)], yticks=[], label="Dow Jones")
fund.plot(y="value", ax=ax, xticks=[0, trainingDays, len(date)], yticks=[], label="Index Fund")
```



Hands-On

Using GAMS Jupyter Notebooks to tell "optimization stories"

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

- This "product" is currently under development.
  Give it a try at https://jupyterhub.gams.com/hub/login

Outlook:
Solving Multiple Models Efficiently

- Solvelink & the GAMS Grid Facility
- GUSS – The Gather-Update-Solve-Scatter

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 solve.

```
Model transport /all/ ;
Option 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 Solves

6_trnsport_solvelink_seq.gms | 6_trnsport_solvelink_seq.lst

```
  Model transport /all/ ;

set s   scenarios / s1*s100 /
    sl solvelink / ChainScript, CallModule, LoadLibrary /;
parameter dd(s,i,j)  distance by scenario
          ff(s)      freight cost by scenario
          time(*)    time for 100 scenarios
          sl_val(sl) solvelink value / ChainScript 0, CallModule 2, LoadLibrary 5 /;
scalar    tmp;

dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
ff(s)     = uniform(0.9,1.1)*f;

option limrow=0, limcol=0, solprint=off;

* SERIAL SOLVE
loop(sl,
  tmp = jnow;
  transport.solvelink=sl_val(sl);
  loop(s,
    d(i,j) = dd(s,i,j);
    f      = ff(s);
    Solve transport using lp minimizing z ;
  );
  time(sl) = (jnow-tmp)*24*60*60;
);
display time;
```

```
----     88 PARAMETER time  time for 100 scenarios
ChainScript 17.042,    CallModule  2.150,    LoadLibrary 0.631
```

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 Solves

7_trnsport_solvelink_async.gms   7_trnsport_solvelink_async.lst

```
 Model transport /all/ ;

option lp=cplexd;
set s  scenarios / s1*s100 /
    sl solvelink / aSyncGrid, aSyncThreads /;
parameter dd(s,i,j)  distance by scenario
          ff(s)      freight cost by scenario
          time(*)    time for 100 scenarios
          sl_val(sl) solvelink value / aSyncGrid 3, aSyncThreads 6 /
          h(s)       scenario handle;
scalar    tmp;

dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
ff(s)     = uniform(0.9,1.1)*f;
option limrow=0, limcol=0, solprint=silent;
* Async SOLVE
loop(sl,
  tmp = jnow;
  transport.solvelink=sl_val(sl);

  loop(s,
    d(i,j) = dd(s,i,j);
    f      = ff(s);
    Solve transport using lp minimizing z ;
    h(s) = transport.handle;    // save instance handle
  );
  repeat
    display$readycollect(h) 'Waiting for next instance to complete';
    loop(s$handlecollect(h(s)),
       display$handledelete(h(s)) 'trouble deleting handles';
       h(s) = 0;    // indicate that we have loaded the solution
    );
  until card(h) = 0 or timeelapsed > 180;  // wait until all models are loaded
  time(sl) = (jnow-tmp)*24*60*60;
);
display time;
```

```
----      97 PARAMETER time  time for 100 scenarios
aSyncGrid     2.991,     aSyncThreads 0.144
```

Solvelink Option – Asynchronous Solves

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

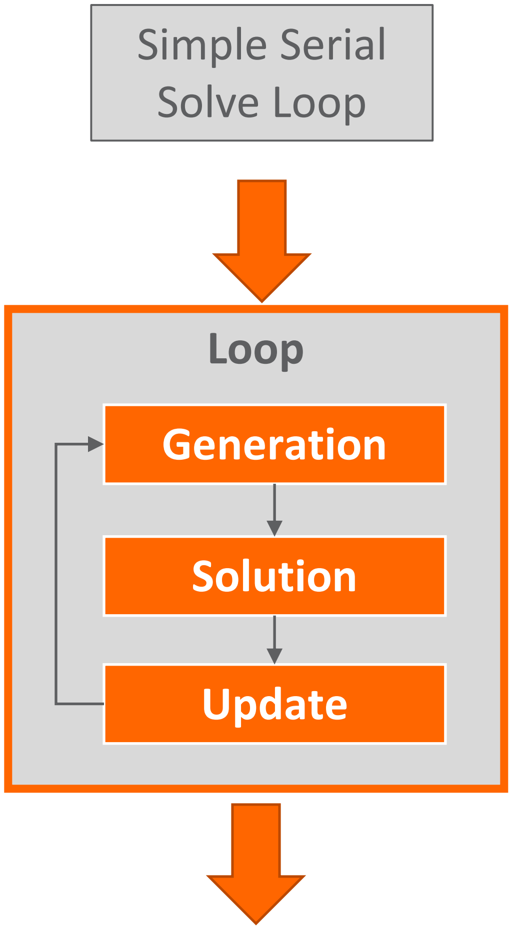
7_dicex_solvelink.gms | 7_dicex_solvelink.lst

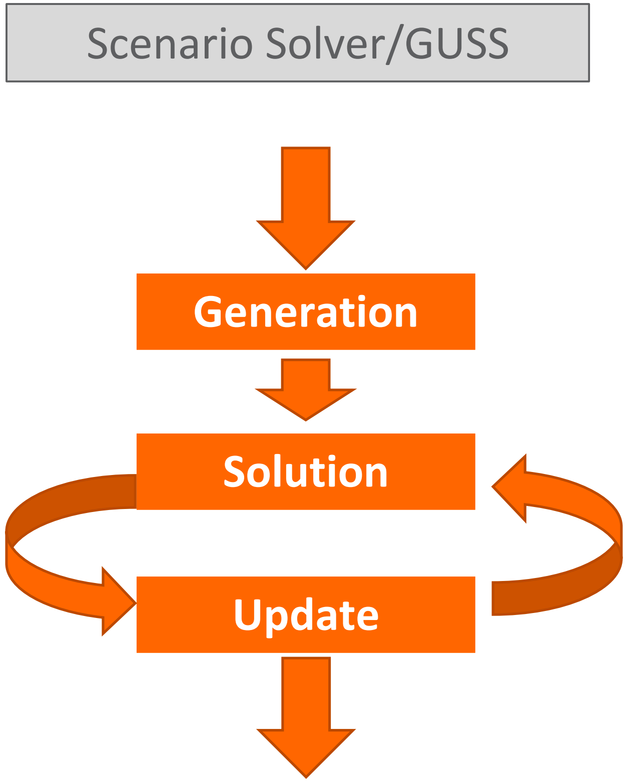
```
option mip=cplexd;
* SEQUENTIAL SOLVE
loop(seq(sl),
  tmp = jnow;
  dice2.solvelink=sl_val(sl);
  loop(s,
    solve dice2 using mip maximizing wnx;
  );
  time(sl) = (jnow-tmp)*24*60*60;
);

* Async SOLVE
loop(async(sl),
  tmp = jnow;
  dice2.solvelink=sl_val(sl);
  loop(s,
    solve dice2 using mip maximizing wnx;
    h(s) = dice2.handle;    // save instance handle
  );
  repeat
    display$readycollect(h) 'Waiting for next instance to complete';
    loop(s$handlecollect(h(s)),
       display$handledelete(h(s)) 'trouble deleting handles';
       h(s) = 0;    // indicate that we have loaded the solution
    );
  until card(h) = 0 or timeelapsed > 180;  // wait until all models are loaded
  time(sl) = (jnow-tmp)*24*60*60;
);
option time:3:0:1;
display time;
```

```
----     124 PARAMETER time
time for 4 scenarios

ChainScript   21.523
CallModule    21.215
LoadLibrary   20.814
aSyncGrid      6.521
aSyncThreads   5.316
```

GUSS – Gather-Update-Solve-Scatter

aka Scenario Solver





Generates model once and updates the algebraic model **keeping the model "hot"** inside the solver.

GUSS – Gather-Update-Solve-Scatter

aka Scenario Solver

8_trnsport_GUSS_solvelink.gms* | 8_trnsport_GUSS_solvelink.lst

```
parameter dd(s,i,j)  distance by scenario
          ff(s)      freight cost by scenario
          time(*)    time for 100 scenarios;
scalar    tmp;

dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
ff(s)     = uniform(0.9,1.1)*f;
option limrow=0, limcol=0, solprint=off;

* GUSS
transport.solvelink = 0;
tmp = jnow;
Set mattrib / system.GUSSModelAttributes /;
Parameter
    xxGUSS(s,i,j)    collector for level of x
    srep(s, mattrib) model attibutes like modelstat etc
    o(*)             GUSS options /  SkipBaseCase 1 /

Set dict   / s . scenario.''
             o . opt     .srep
             d . param   .dd
             f . param   .ff
             x . level   .xxGUSS /

Solve transport using lp minimizing z scenario dict;
time('GUSS') = (jnow-tmp)*24*60*60;

display time;
```

```
----     137 PARAMETER time
time for 100 scenarios

ChainScript   4.65
CallModule    1.80
LoadLibrary   0.44
aSyncGrid     3.22
aSyncThreads  3.18
GUSS          0.36
```

Grid & GUSS – Examples from the model library

- trnsgrid: 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
  - Asynchronous solves in combined submission & collection loop. Keep number of submitted models <= #threads
- gussgrid: 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

GAMS and High Performance Computing
Thursday, September 13, 9:00 am - 10:15 am
(Session TA-5)

Thank You

Meet us at the GAMS booth!