GAMS

GAMS – An Introduction

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

Agenda

GAMS at a Glance

GAMS – Hands On Examples

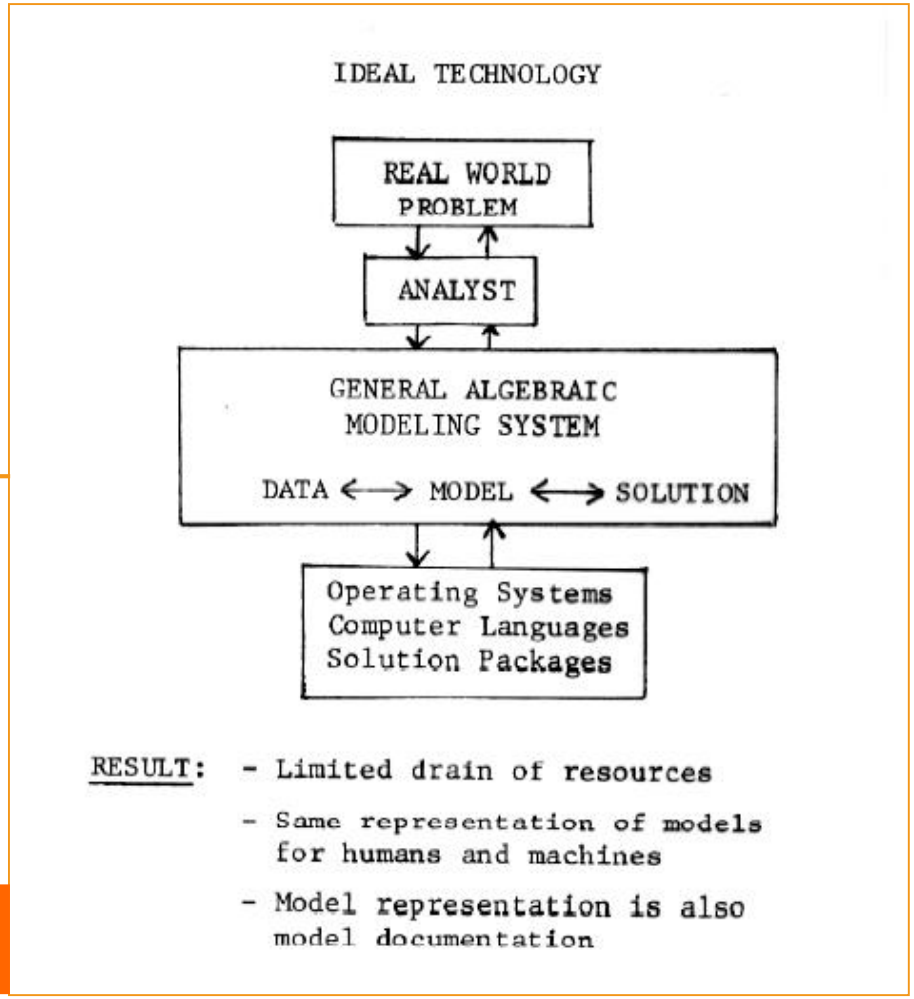
Outlook

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

GAMS at a Glance

1976 - A World Bank Slide

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. Bisschop, 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]

What did this give us?

Simplified model development & maintenance

Increased productivity tremendously

Made mathematical optimization available to a broader audience (domain experts)

➢ 2012 INFORMS Impact Prize

Company

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

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

Foundation of GAMS

Powerful algebraic modeling language

Open architecture and interfaces to other systems, independent layers

Declarative and Procedural Language Elements

Declarative elements

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

Procedural elements

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

Independence of Model and Operating System

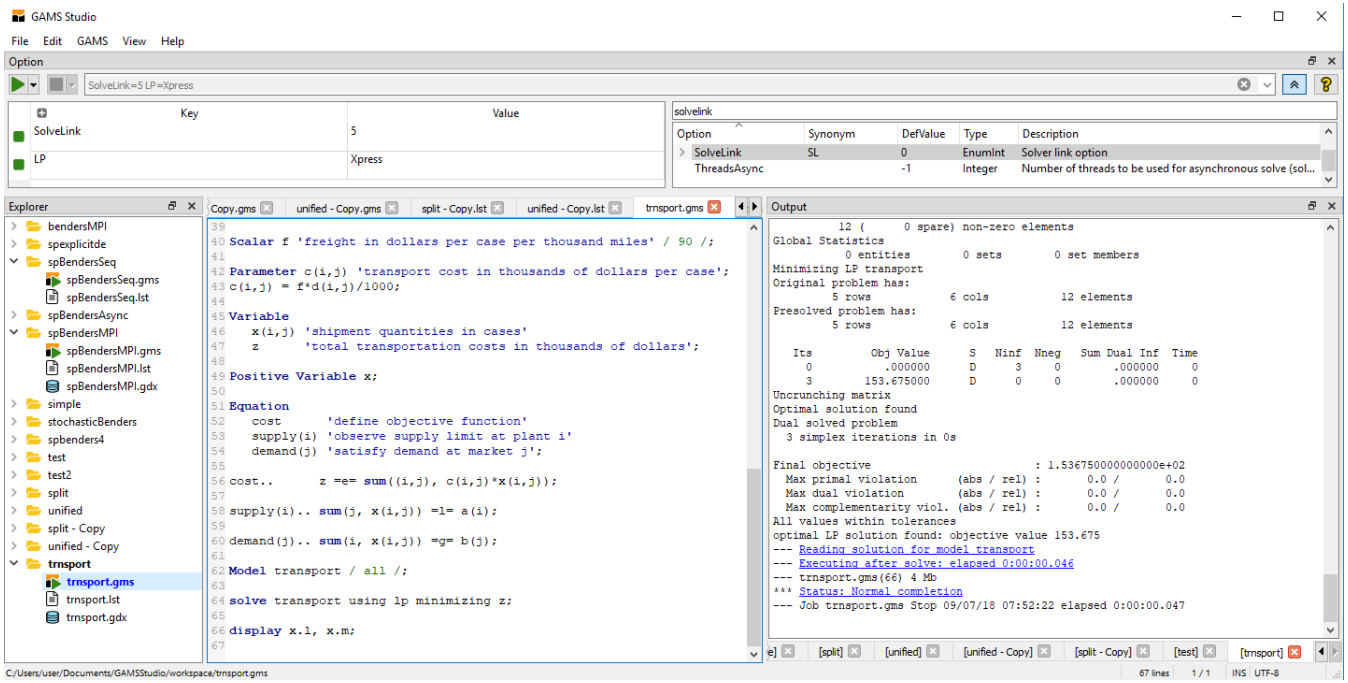
Platforms supported by GAMS:

➡Models can be moved between platforms with ease!

Cross Platform GUI – GAMS Studio

- Open source Qt project (Mac/Linux/Win)
  - Published on GitHub under GPL
- Released in beta status
- Group Explorer
- Editor / Syntax coloring / Spell checks

- Tree view / Syntax-error navigation
- Option Editor
- Integrated Help
- Model Debugging & Profiling
- Solver selection & setup
- Data viewer
- GAMS Processes Control



Independence of Model and Solver

One environment for a wide range of model types and solvers

All major commercial LP/MIP solver

Open Source Solver (COIN)

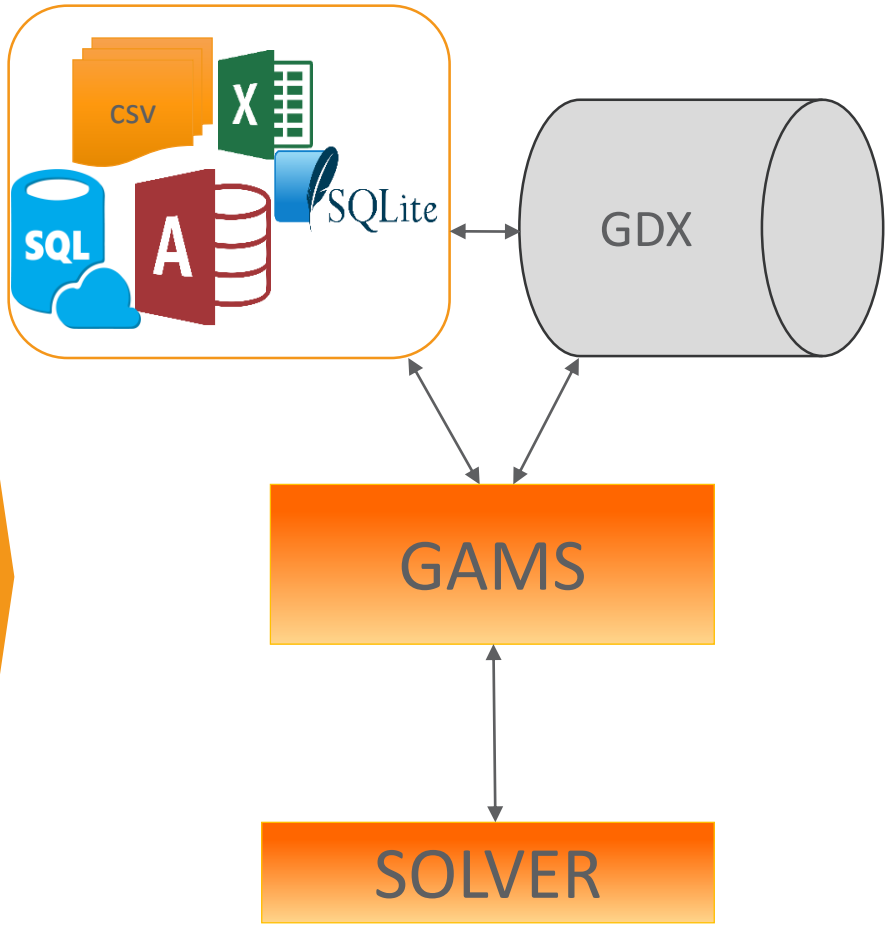
Also solver for NLP, MINLP, global, and stochastic optimization

Switching between solvers with one line of code!

Independence of Model and Data

- Declarative Modeling
- ASCII: Initial model development

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

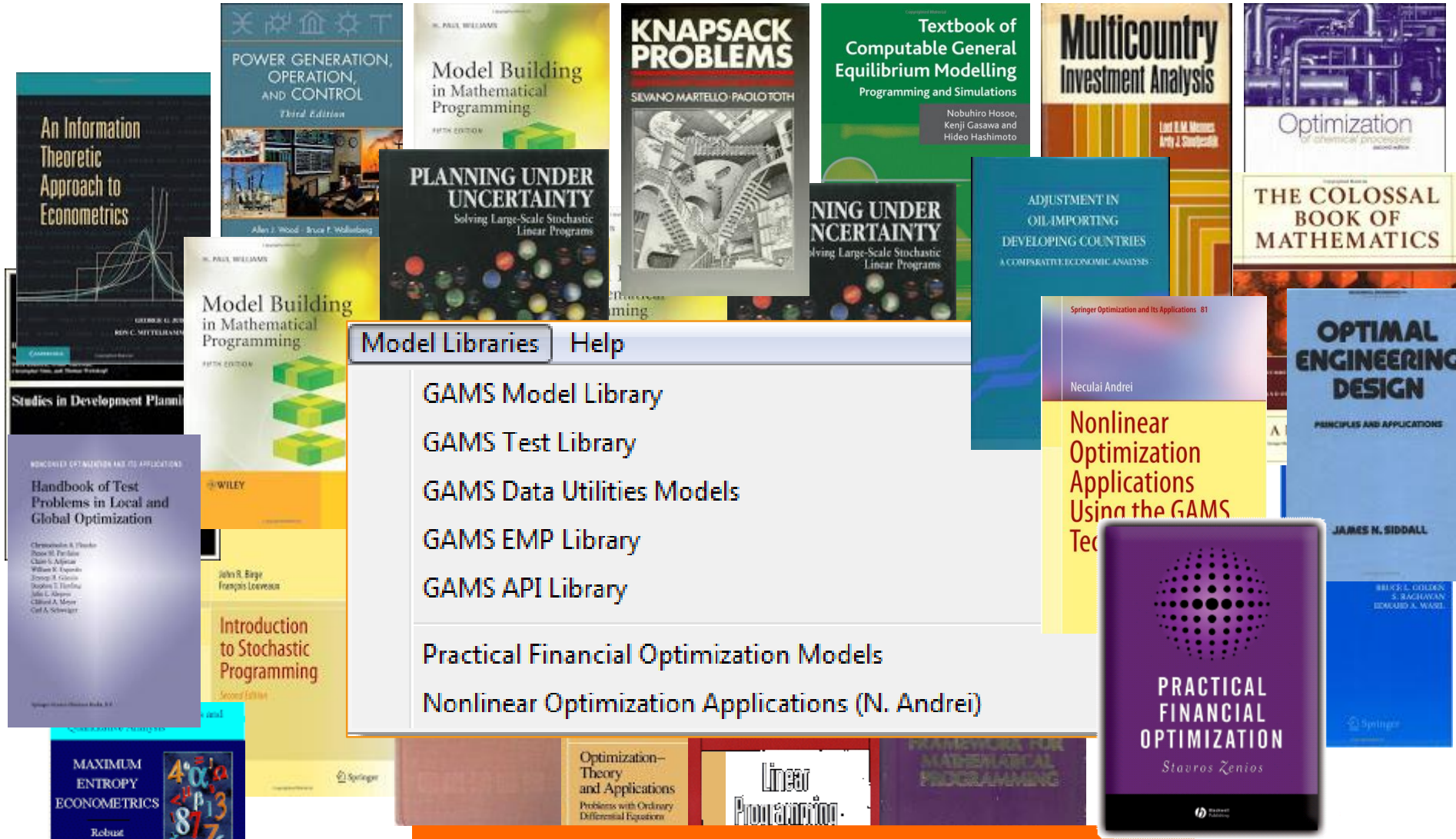


Independence of Model and User Interface

API's

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

Free Model Libraries



➢ **More than 1400 models!**

Why GAMS?

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

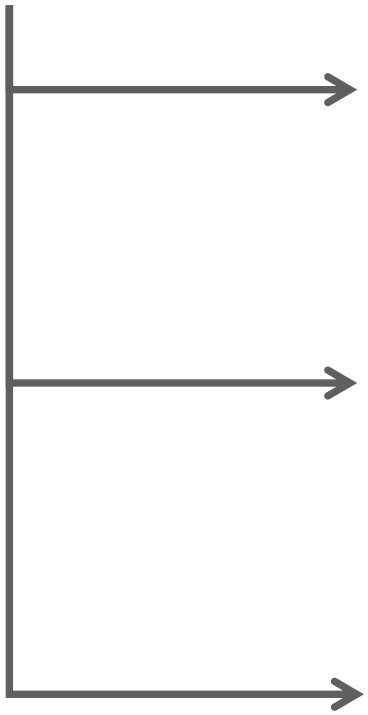
- Consistent implementation of design principles
  - Simple, but powerful modeling language
  - Independent layers
  - Open architecture: Designed to interact with other applications

- Open for new developments
- Protecting investments of users

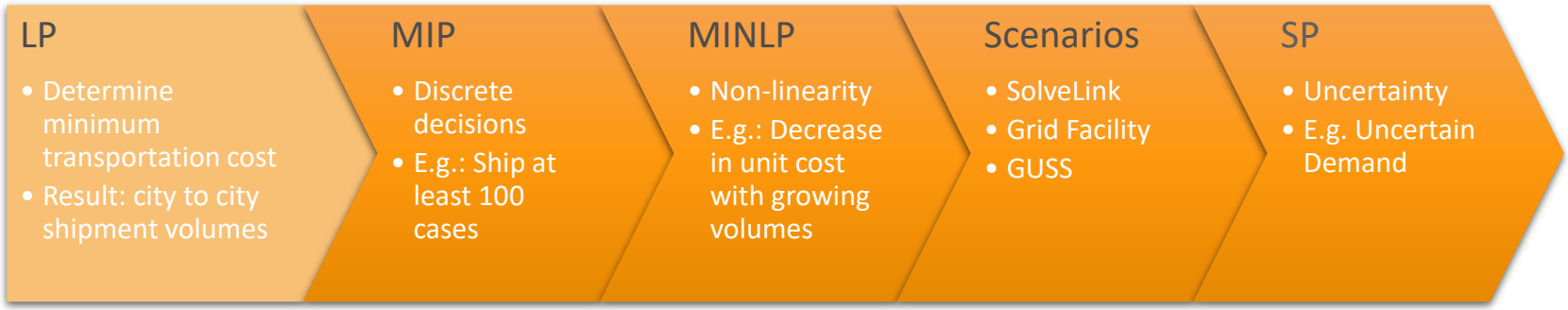
GAMS –
Hands On Examples

A Simple Transportation Problem

- What does this example show?

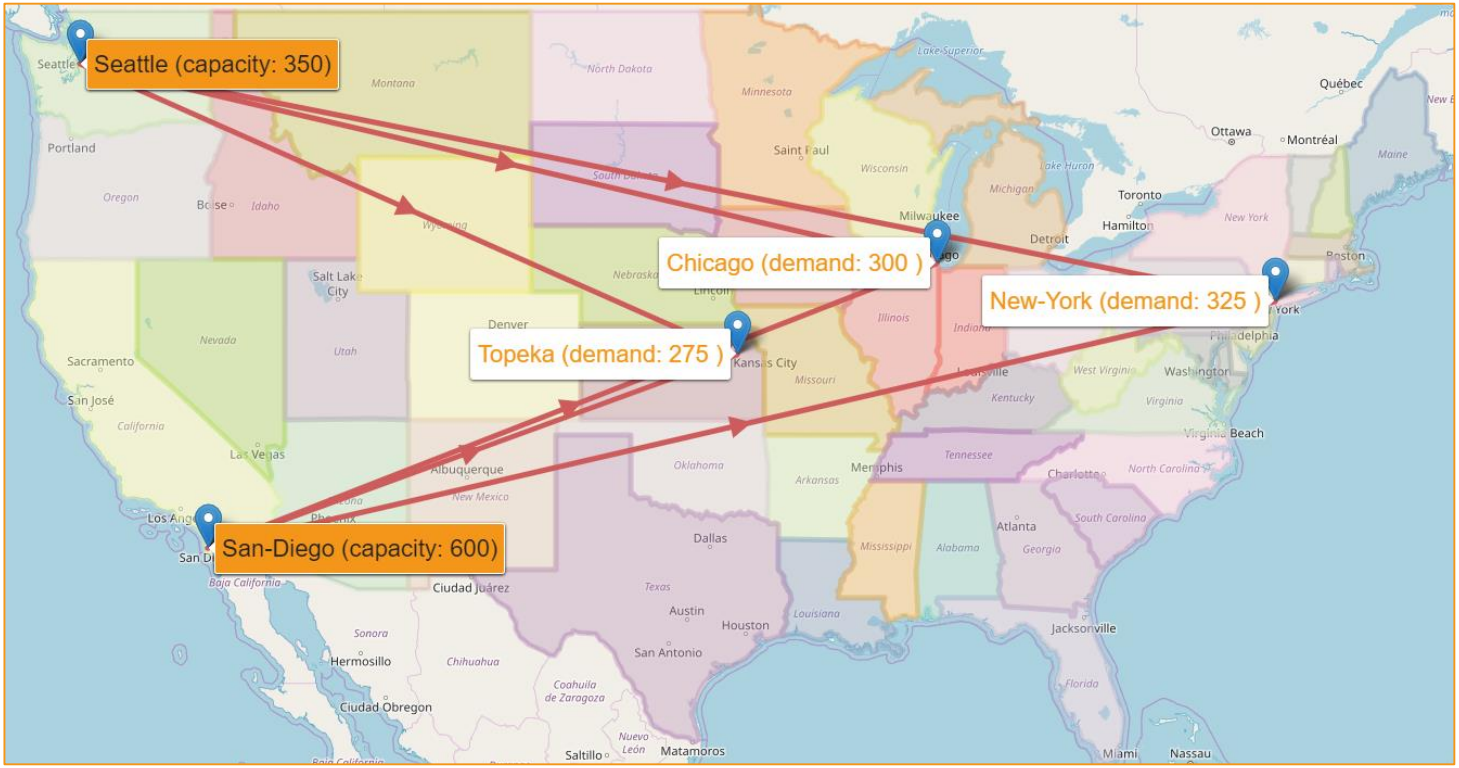


- It gives a first glimpse of how a problem can be formulated in GAMS
- It shows some basics of data exchange with GAMS
- It shows how easy it is to change model type and, consequently, solver technology


LP
• Determine minimum transportation cost
• Result: city to city shipment volumes
MIP
• Discrete decisions
• E.g.: Ship at least 100 cases
MINLP
• Non-linearity
• E.g.: Decrease in unit cost with growing volumes
Scenarios
• SolveLink
• Grid Facility
• GUSS
SP
• Uncertainty
• E.g. Uncertain Demand

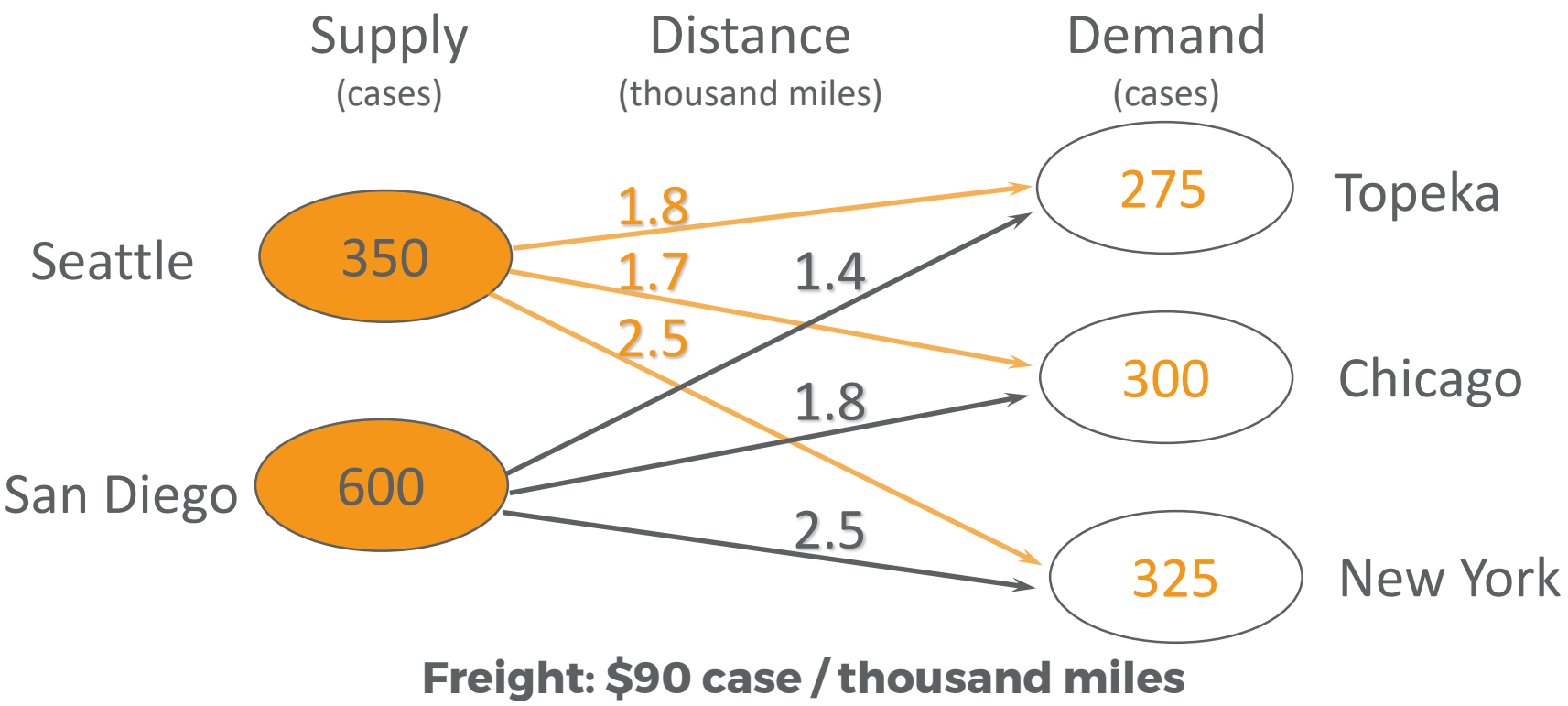
A Simple Transportation Problem

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



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)

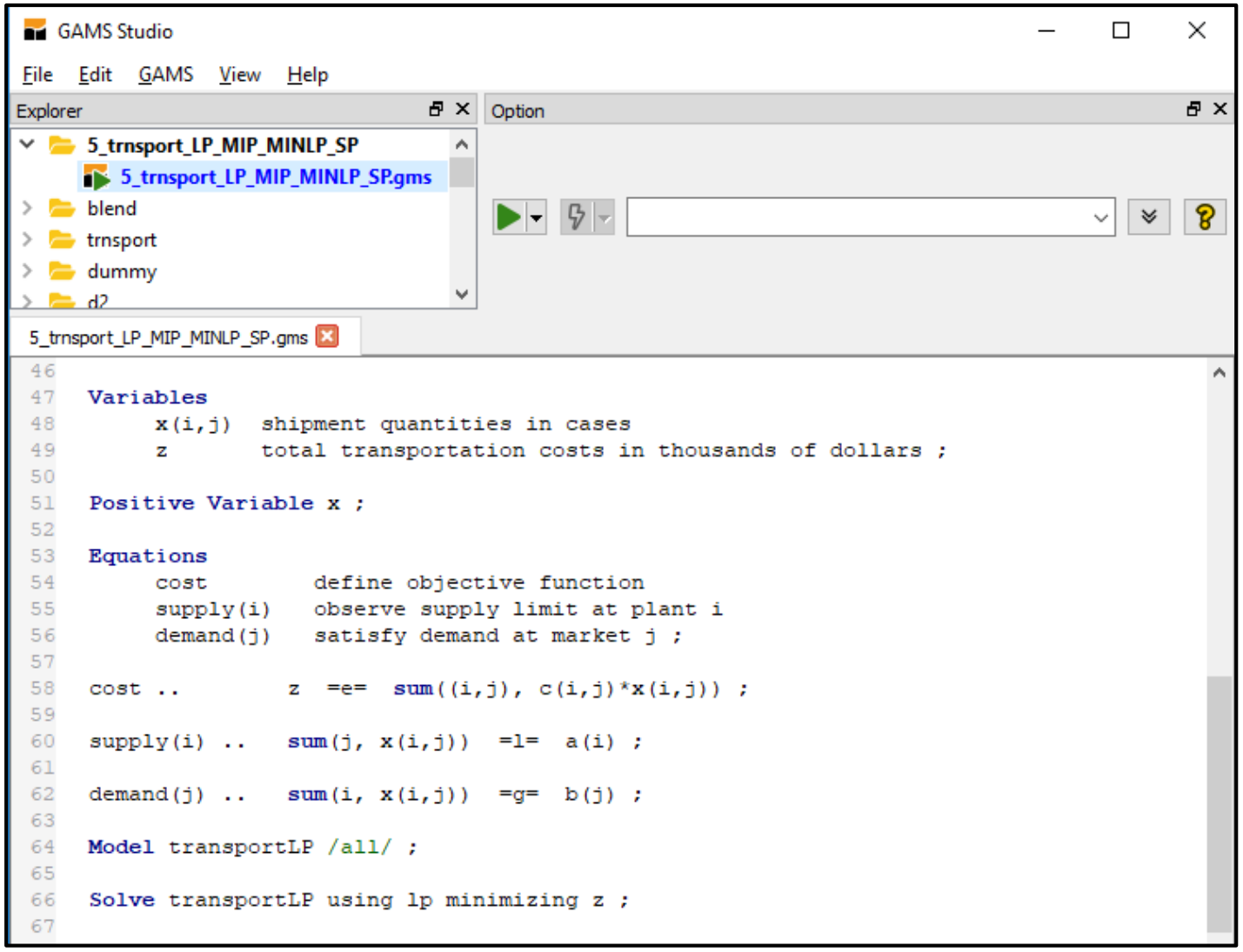
$\sum_i x_{ij} \geq b_j \quad \forall\, j$ (Shipments to each market $\geq$ demand)

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

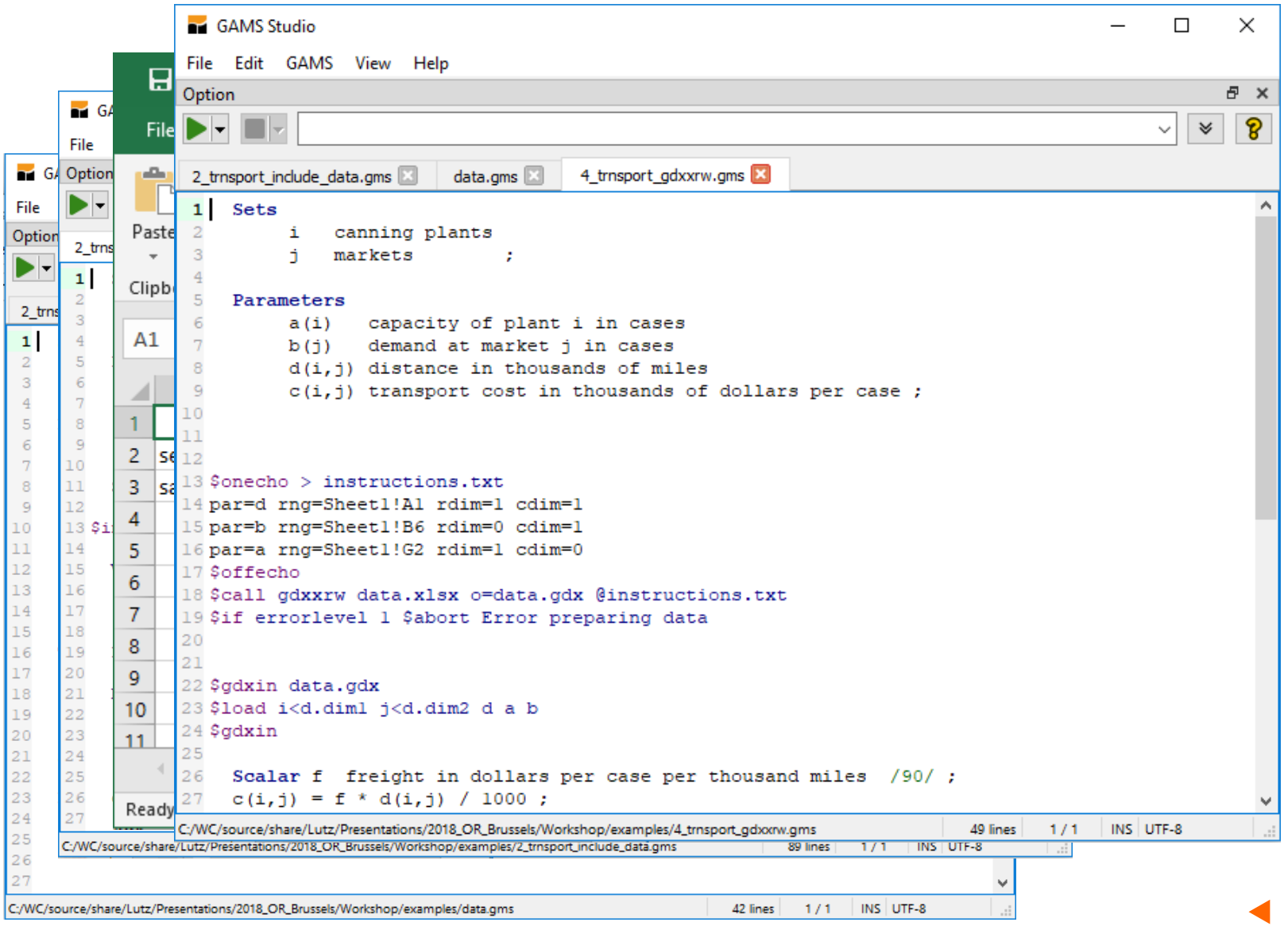
$i, j \in \mathbb{N}$

Hands-On

GAMS Syntax (LP Model)

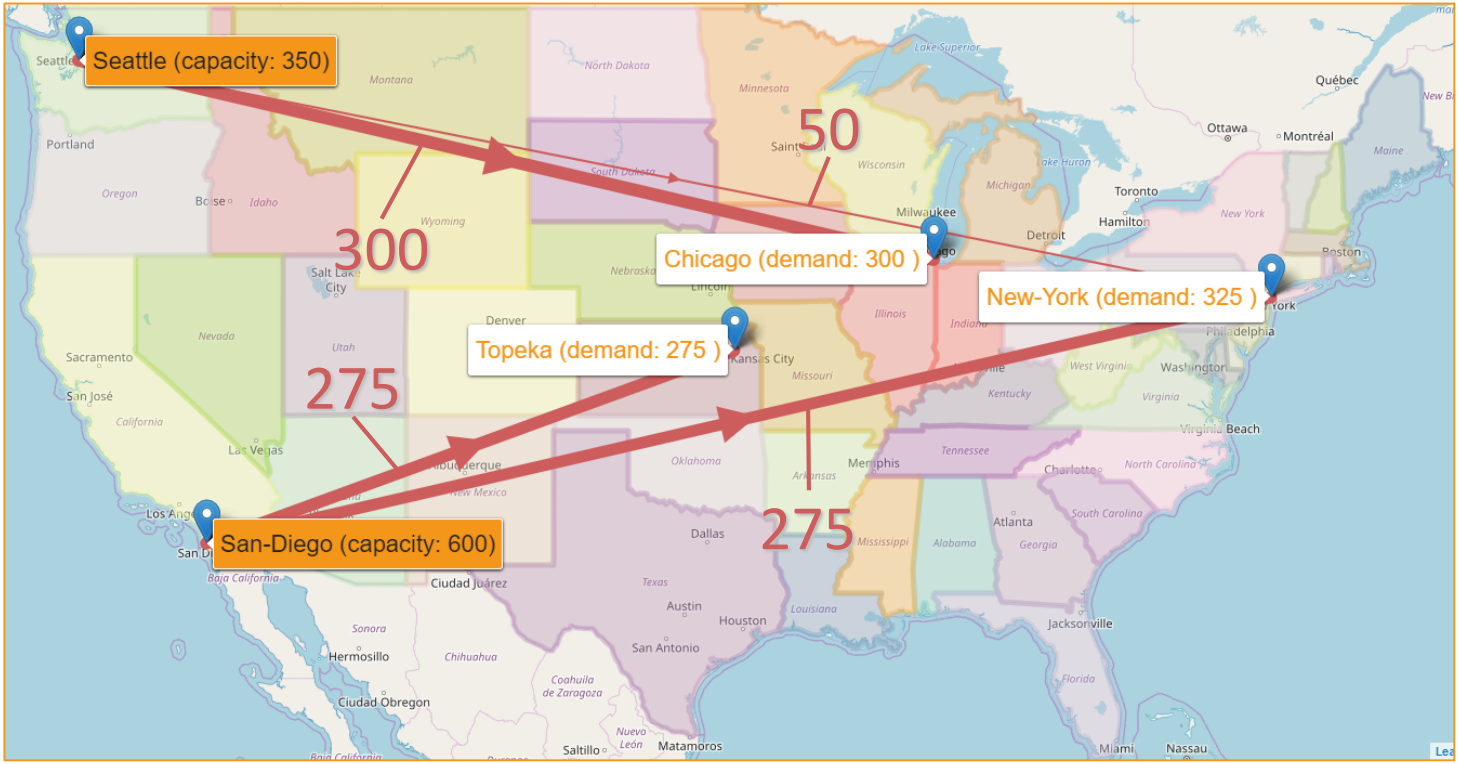


GAMS Syntax (Data Input)



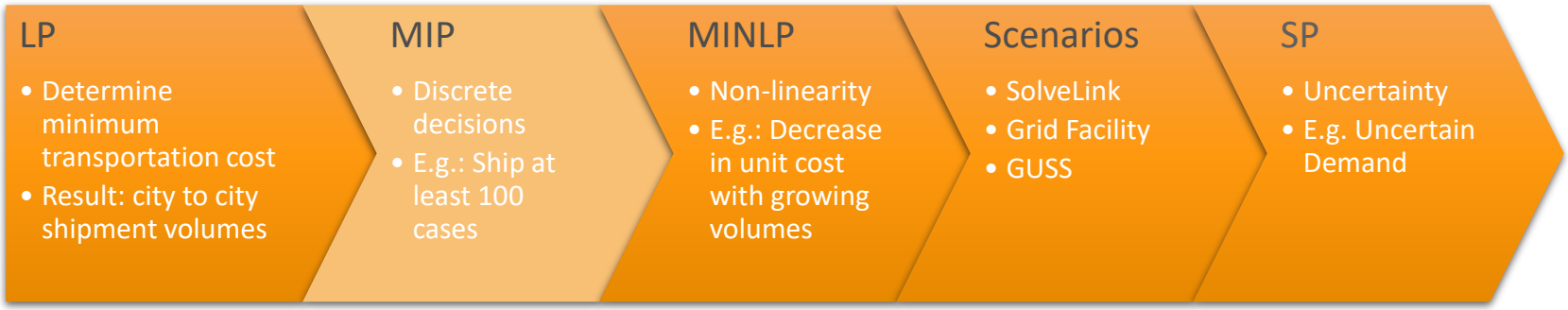
Solution to LP model

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



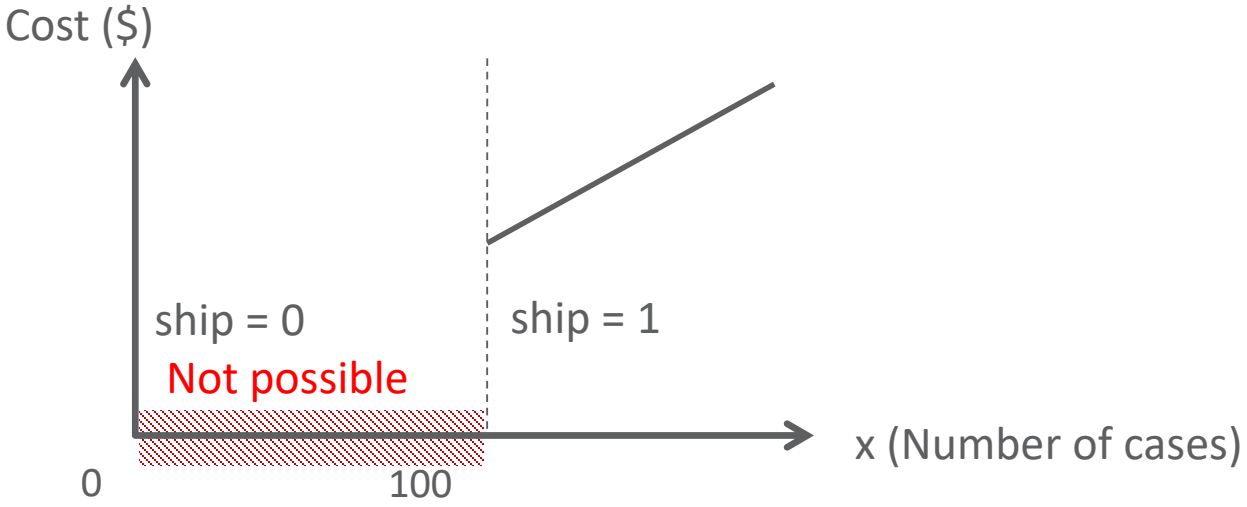
Freight: $90 case / thousand miles

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
Scenarios
• SolveLink
• Grid Facility
• GUSS
SP
• Uncertainty
• E.g. Uncertain Demand

MIP Model:
Minimum Shipment of 100 cases

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



add constraints:

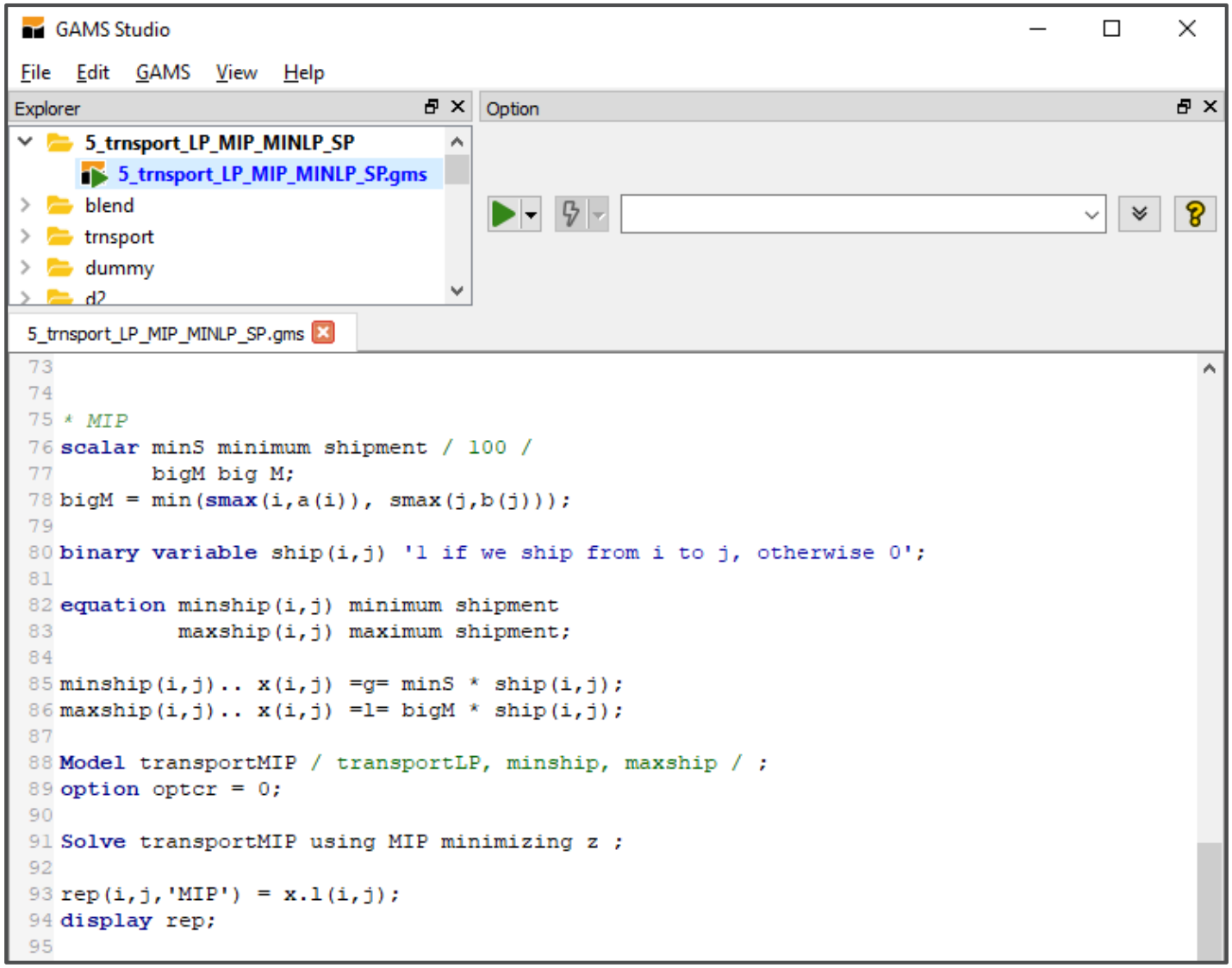
$x_{i,j} \geq 100 \cdot ship_{i,j} \quad \forall\, i,j$ (if ship=1, then ship at least 100)

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

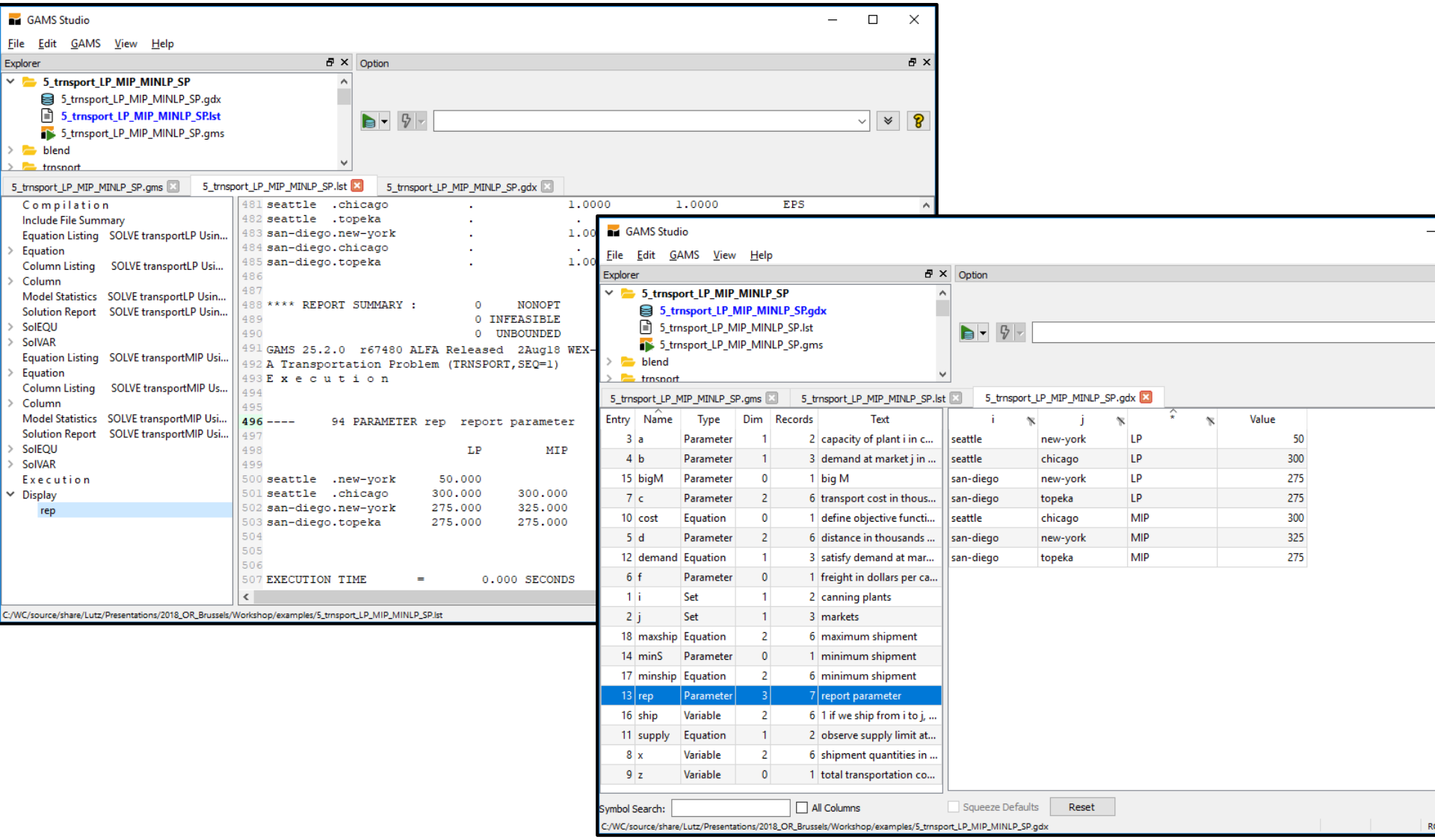
$ship_{i,j} \in \{0,1\}$

Hands-On

MIP Model: GAMS Syntax

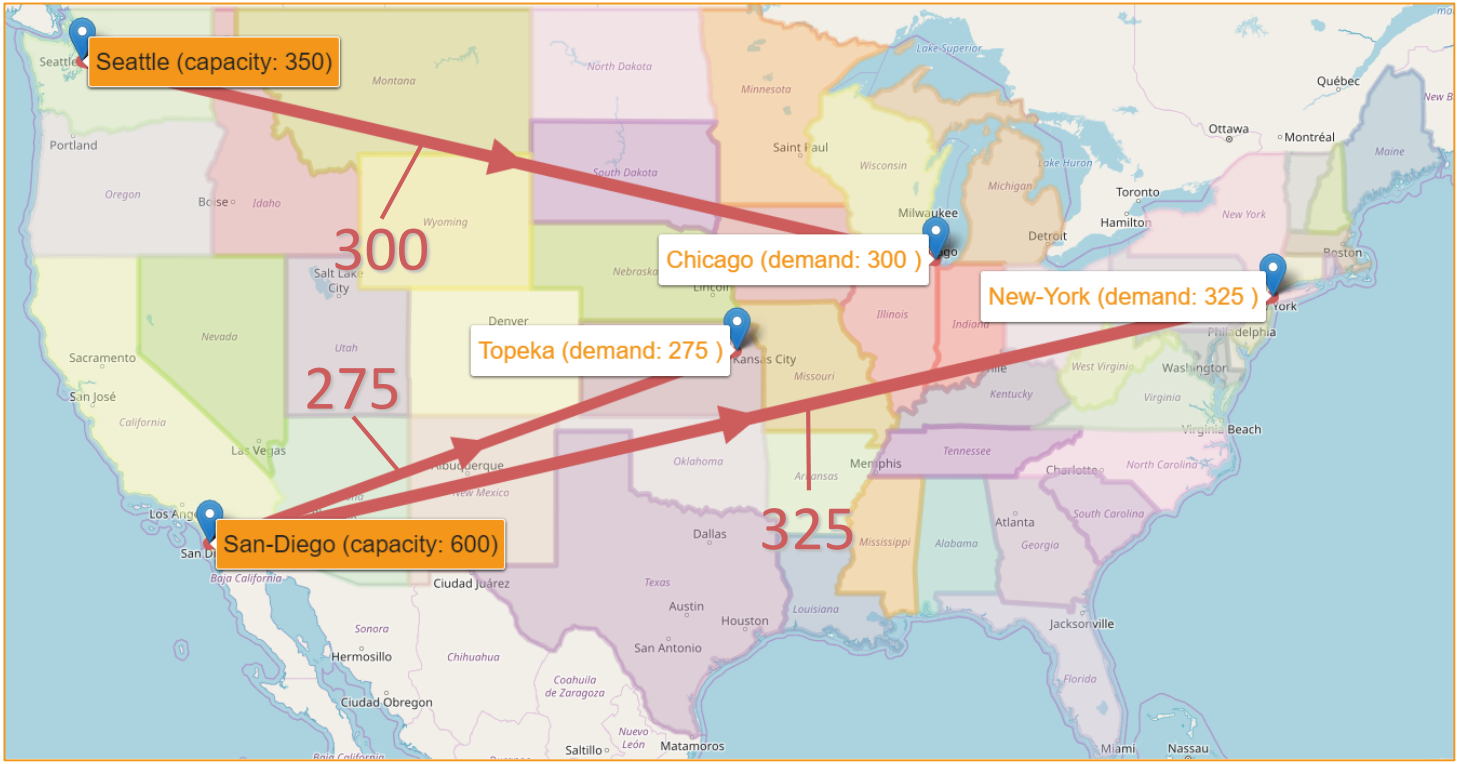


MIP Model: Results



MIP Model: Solution

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

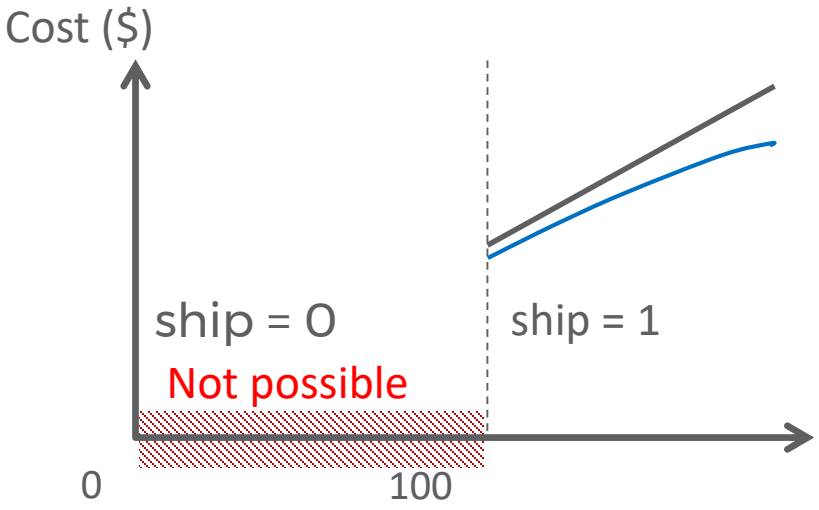


Freight: $90 case / thousand miles

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
Scenarios
• SolveLink
• Grid Facility
• GUSS
SP
• Uncertainty
• E.g. Uncertain Demand

MINLP: Cost Savings



The cost per case decreases with an increasing shipment volume

Replace:

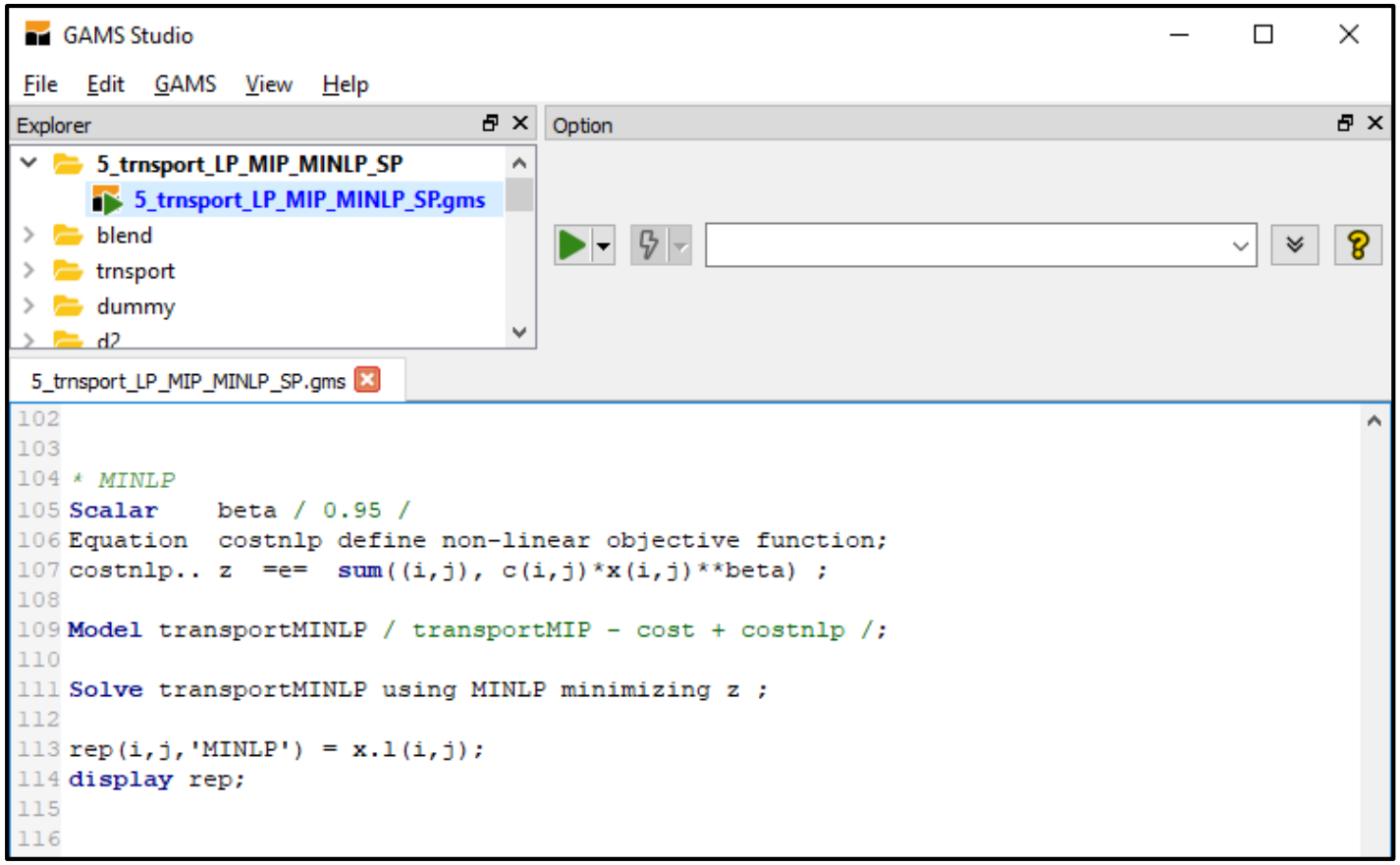
min $\sum_i \sum_j c_{ij} \cdot x_{ij}$ (Minimize total transportation cost)

With

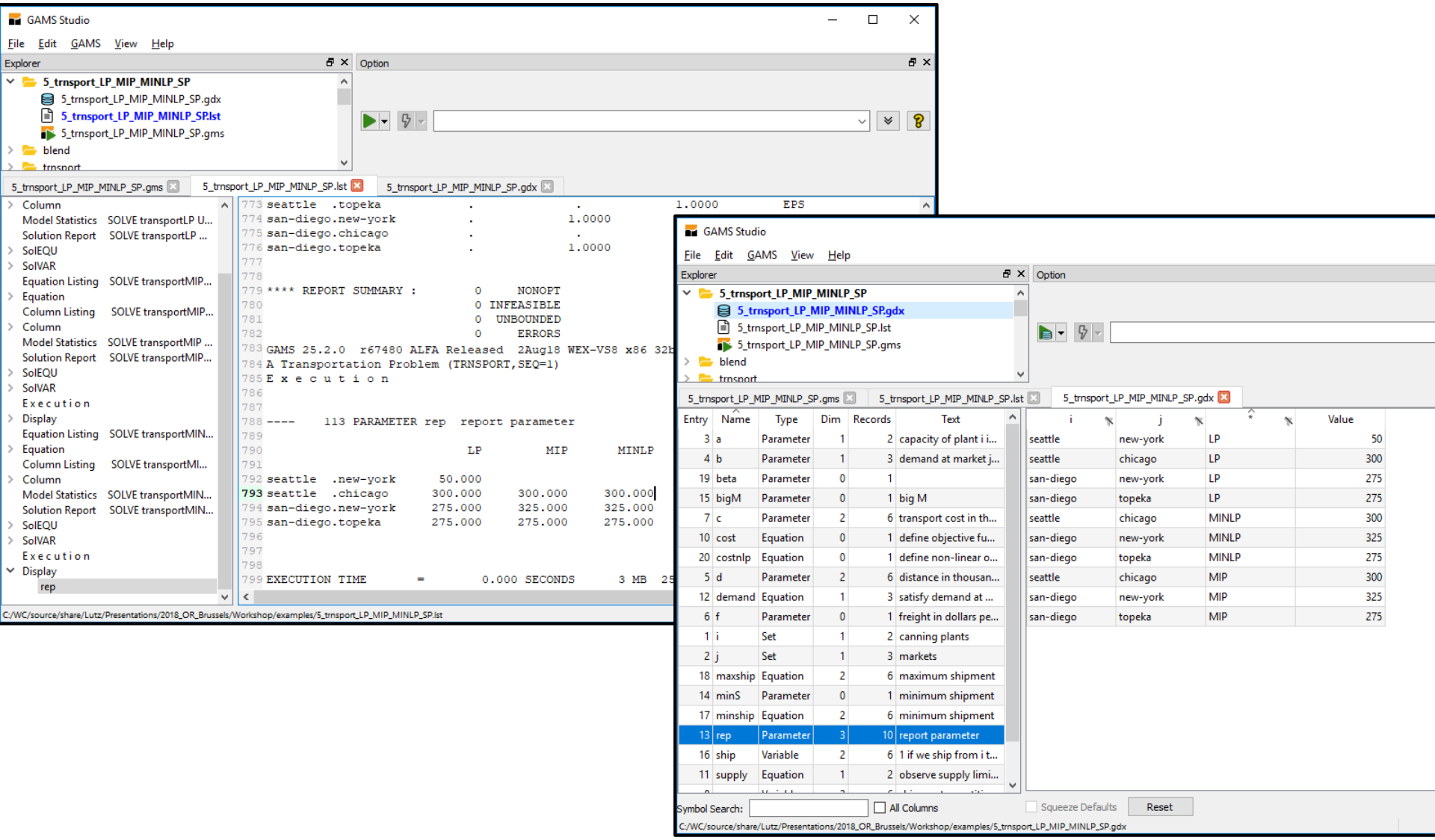
min $\sum_i \sum_j c_{ij} \cdot x_{ij}^{beta}$ (Minimize total transportation cost)

Hands-On

MINLP Model: GAMS Syntax

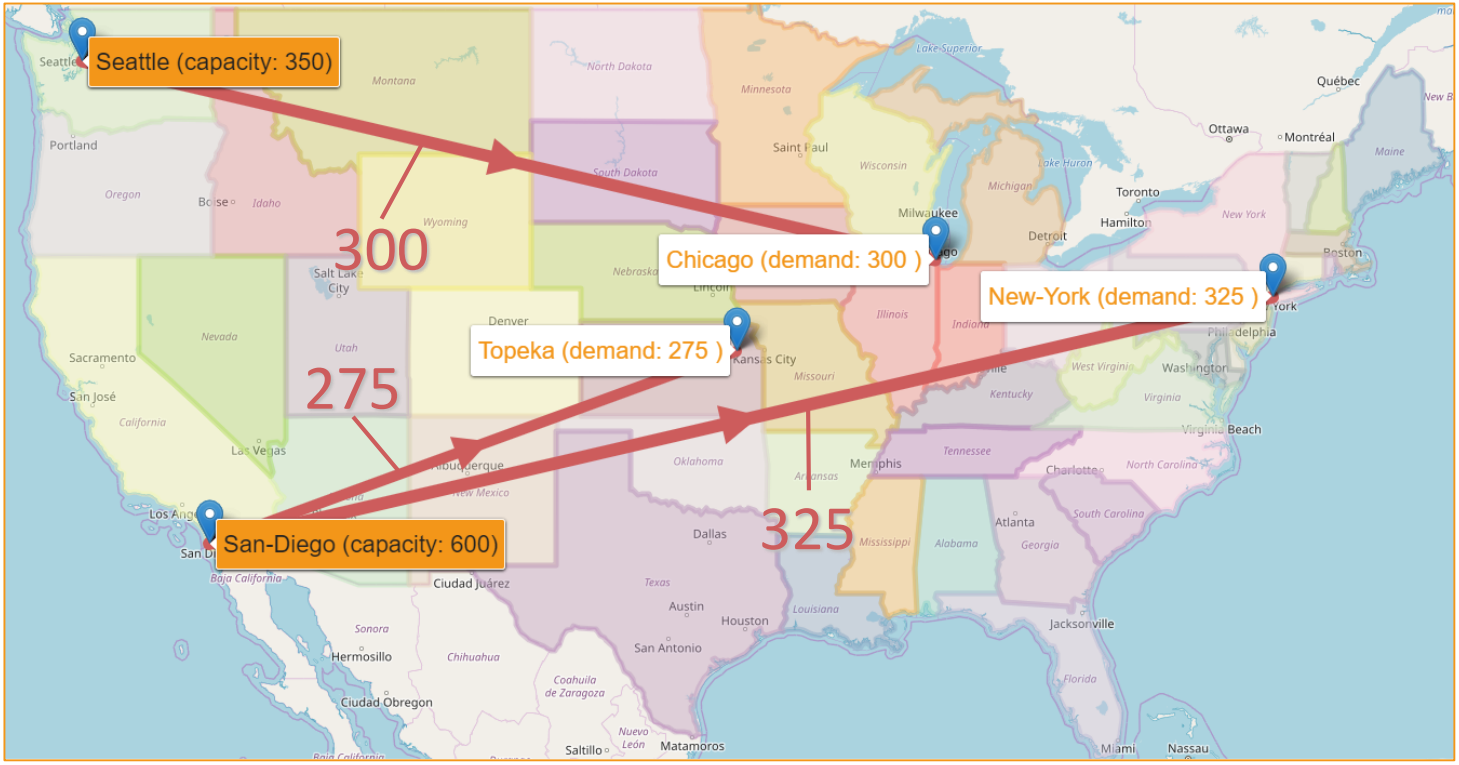


MINLP Model: Results



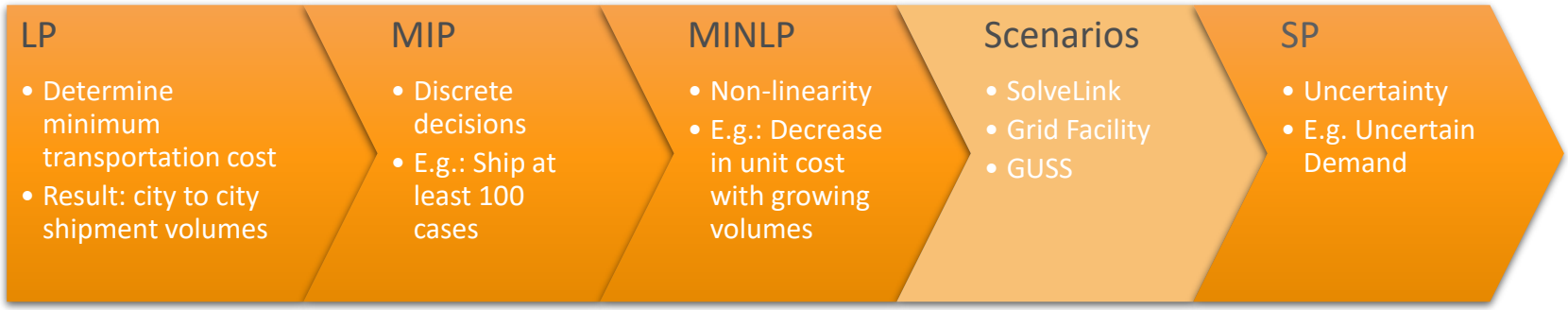
MINLP Model: Solution

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
Scenarios
• SolveLink
• Grid Facility
• GUSS
SP
• Uncertainty
• E.g. Uncertain Demand

Motivation

- Solving challenging real-world problems often involves the solution of many optimization problems
  - Decomposition Methods
  - Scenario Analysis
  - Heuristics
  - …
- Such approaches are often chosen, if solving the problem at hand does not work with a single monolithic model, e.g.
  - Due to it's size and the required resources (e.g. memory)
  - Due to time restrictions (Problem should be solved in minutes but it takes days)
  - …

→GAMS Grid Facility

→Gather-Update-Solve-Scatter

SolveLink Option

Controls GAMS function when linking to solver.

```
Model transport /all/ ;
transport.solveLink={0   %Solvelink.ChainScript%,
                     1   %Solvelink.CallScript%,
                     2   %Solvelink.CallModule%,
                     3   %Solvelink.AsyncGrid%,
                     4   %Solvelink.AsyncSimulate%,
                     5   %Solvelink.LoadLibrary%,
                     6   %solveLink.aSyncThreads%,
                     7   %solveLink.threadsSimulate%};
solve transport using lp minimizing z;
```

SolveLink Option – Sequential Solves

- ChainScript [0]: Solver process, GAMS vacates memory
  - + Maximum memory available to solver
  - + protection against solver failure (*hostile* link)
  - - swap to disk

- Call{Script [1]/Module [2]}: Solver process, GAMS stays live
  - + protection against solver failure (*hostile* link)
  - + no swap of GAMS database
  - - file based model communication

- LoadLibrary [5]: Solver DLL in GAMS process
  - + fast memory based model communication
  - + update of model object inside the solver (hot start)
  - - not supported by all solvers

SolveLink Option Sequential – Exercise

- Generate 100 distance scenarios with random data
  - Hint: Look, e.g., at the GAMS function **`uniform`**
- Solve these scenarios with the solveLink values…
  - ChainScript [0]
  - CallModule [2]
  - LoadLibrary [5]
- Compare the execution time of solving all scenarios with different solveLink settings
  - Hint: Look at the GAMS function **`jNow`**

Hands-On

SolveLink Option – Sequential Solves

Welcome | 6_trnsport_solvelink_seq.gms | 6_trnsport_solvelink_seq.lst

```
64   Model transport /all/ ;
65
66 set s  scenarios / s1*s100 /
67     sl solvelink / ChainScript, CallModule, LoadLibrary /;
68 parameter dd(s,i,j)  distance by scenario
69           time(*)    time for 100 scenarios
70           sl_val(sl) solvelink value / ChainScript %solveLink.chainScript%,
71                                        CallModule  %solveLink.CallModule%,
72                                        LoadLibrary %solveLink.loadLibrary% /;
73 scalar    tmp;
74
75 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
76
77 option limrow=0, limcol=0, solprint=silent;
78
79 * SERIAL SOLVE
80 loop(sl,
81   tmp = jnow;
82   transport.solvelink=sl_val(sl);
83   loop(s,
84     d(i,j) = dd(s,i,j);
85     Solve transport using lp minimizing z ;
86   );
87   time(sl) = (jnow-tmp)*24*60*60;
88 );
89 display time;
90
```

```
----     88 PARAMETER time  time for 100 scenarios
ChainScript 6.710,    CallModule  2.694,    LoadLibrary 0.578
```

SolveLink Option – Asynchronous Solves

- aSyncGrid [3]: GAMS starts the solution and continues in a Grid computing environment
- aSyncThreads [6]: The problem is passed to the solver in core without use of temporary files, GAMS does not wait for the solver to come back

SolveLink Option Asynchronous – Exercise

- Generate 100 distance scenarios with random data
- Solve these scenarios with the solveLink values…
  - aSyncGrid [3]
  - aSyncThreads [6]
- Compare the execution time of solving all scenarios with different solveLink settings
  - Hint: Check the log for output about solveLink
    - → Use solver CplexD instead of Cplex
  - Hint: Look at the following GAMS functions:
    - **`readyCollect`**
    - **`handleCollect`**
    - **`handleDelete`**

Hands-On

SolveLink Option – Asynchronous Solves

Welcome | 6_trnsport_solvelink_seq.gms | 6_trnsport_solvelink_seq.lst | 7_trnsport_solvelink_async.gms | 7_trnsport_solvelink_async.lst

```
64   Model transport /all/ ;
65
66 set s  scenarios / s1*s100 /
67     sl solvelink / aSyncGrid, aSyncThreads /;
68 parameter dd(s,i,j)  distance by scenario
69           time(*)    time for 100 scenarios
70           sl_val(sl) solvelink value / aSyncGrid    %solveLink.aSyncGrid%,
71                                        aSyncThreads %solveLink.aSyncThreads% /
72           h(s)       scenario handle;
73 scalar    tmp;
74
75 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
76 option limrow=0, limcol=0, solprint=silent, lp=cplexd;
77 * Async SOLVE
78 loop(sl,
79   tmp = jnow;
80   transport.solvelink=sl_val(sl);
81
82   loop(s,
83     d(i,j) = dd(s,i,j);
84     Solve transport using lp minimizing z ;
85     h(s) = transport.handle;    // save instance handle
86   );
87   repeat
88     display$readycollect(h) 'Waiting for next instance to complete';
89     loop(s$handlecollect(h(s)),
90        display$handledelete(h(s)) 'trouble deleting handles';
91        h(s) = 0;    // indicate that we have loaded the solution
92     );
93   until card(h) = 0 or timeelapsed > 180;  // wait until all models are loaded
94   time(sl) = (jnow-tmp)*24*60*60;
95 );
96 display time;
97
```

ChainScript	6.710
CallModule	2.694
LoadLibrary	0.578
aSyncGrid	4.259
aSyncThreads	0.496

```
----     96 PARAMETER time  time for 100 scenarios
aSyncGrid    4.259,    aSyncThreads 0.496
```

SolveLink Option – Asynchronous Solves

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

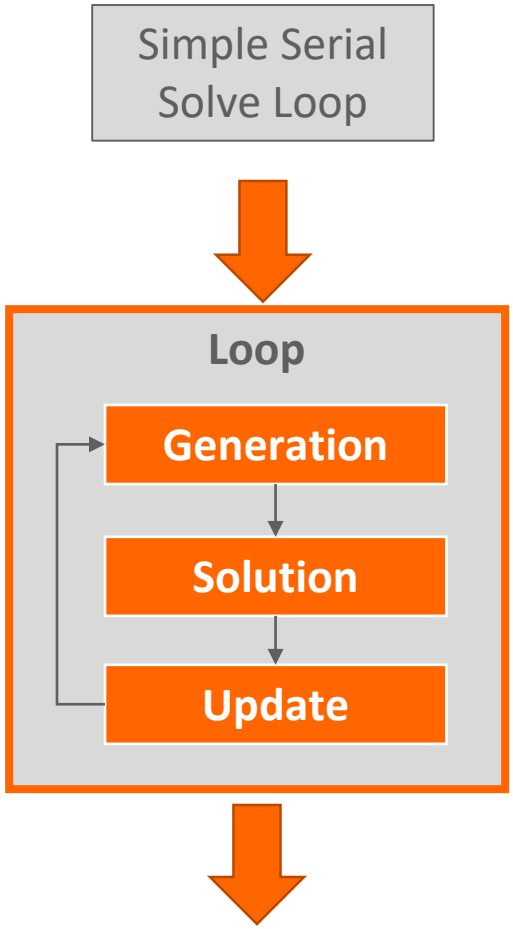
7_dicex_solvelink.gms | 7_dicex_solvelink.lst

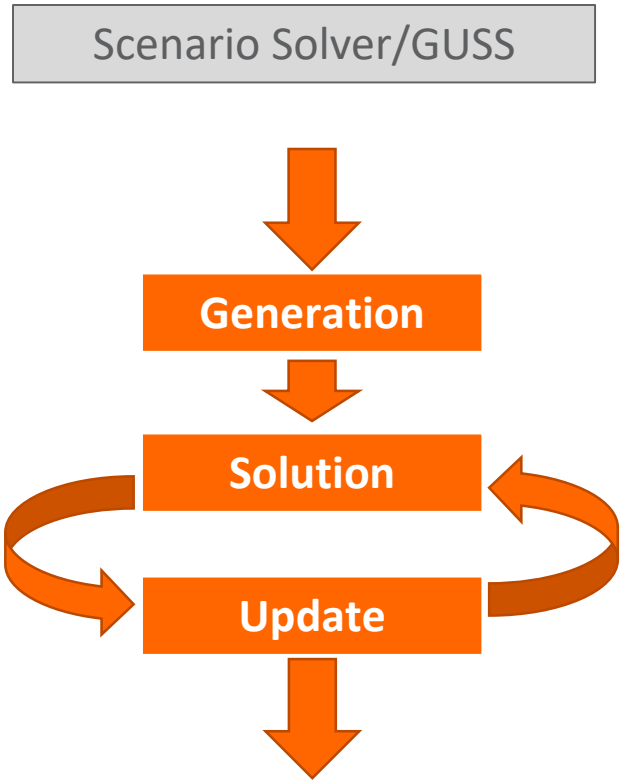
```
95
96 * SEQUENTIAL SOLVE
97 loop(seq(sl),
98   tmp = jnow;
99   dice2.solvelink=sl_val(sl);
100  loop(s,
101    solve dice2 using mip maximizing wnx;
102  );
103  time(sl) = (jnow-tmp)*24*60*60;
104 );
105
106 * Async SOLVE
107 loop(async(sl),
108   tmp = jnow;
109   dice2.solvelink=sl_val(sl);
110   loop(s,
111     solve dice2 using mip maximizing wnx;
112     h(s) = dice2.handle;    // save instance handle
113   );
114   repeat
115     display$readycollect(h) 'Waiting for next instance to complete';
116     loop(s$handlecollect(h(s)),
117        display$handledelete(h(s)) 'trouble deleting handles';
118        h(s) = 0;    // indicate that we have loaded the solution
119     );
120   until card(h) = 0 or timeelapsed > 180;  // wait until all models are loaded
121   time(sl) = (jnow-tmp)*24*60*60;
122 );
123 option time:3:0:1;
124 display time;
```

```
ChainScript   29.807
CallModule    30.004
LoadLibrary   28.864
aSyncGrid      9.112
aSyncThreads   7.901
```

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

```
69 parameter dd(s,i,j)  distance by scenario
70           ff(s)      freight cost by scenario
71           time(*)    time for 100 scenarios;
72 scalar    tmp;
73
74 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
75 ff(s)     = uniform(0.9,1.1)*f;
76 option limrow=0, limcol=0, solprint=off;
77
78 * GUSS
79 transport.solvelink = 0;
80 tmp = jnow;
81 Set mattrib / system.GUSSModelAttributes /;
82 Parameter
83     xxGUSS(s,i,j)    collector for level of x
84     srep(s, mattrib) model attibutes like modelstat etc
85     o(*)             GUSS options /  SkipBaseCase 1 /
86
87 Set dict   / s . scenario.''
88              o . opt      .srep
89              d . param    .dd
90              f . param    .ff
91              x . level    .xxGUSS /
92
93 Solve transport using lp minimizing z scenario dict;
94 time('GUSS') = (jnow-tmp)*24*60*60;
95
96 display time;
97
```

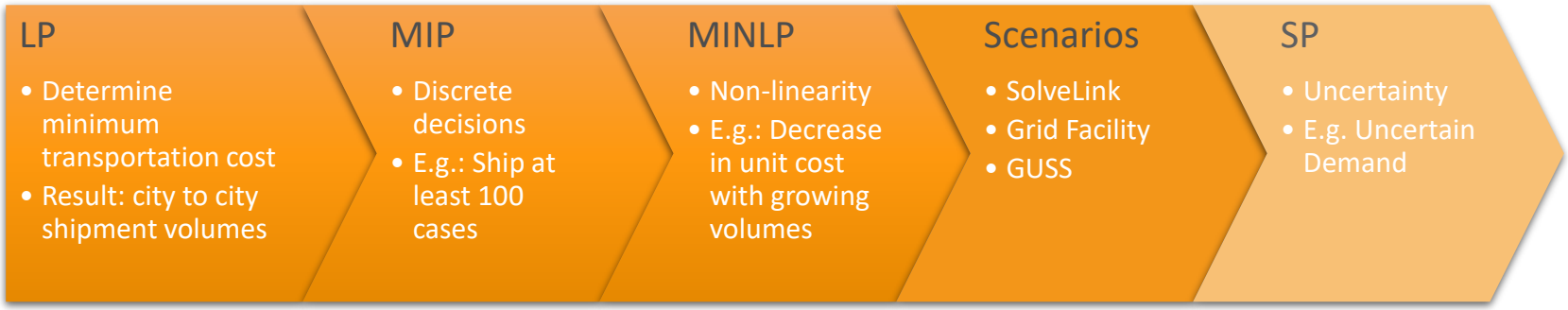
ChainScript	6.710
CallModule	2.694
LoadLibrary	0.578
aSyncGrid	4.259
aSyncThreads	0.496
GUSS	0.273

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

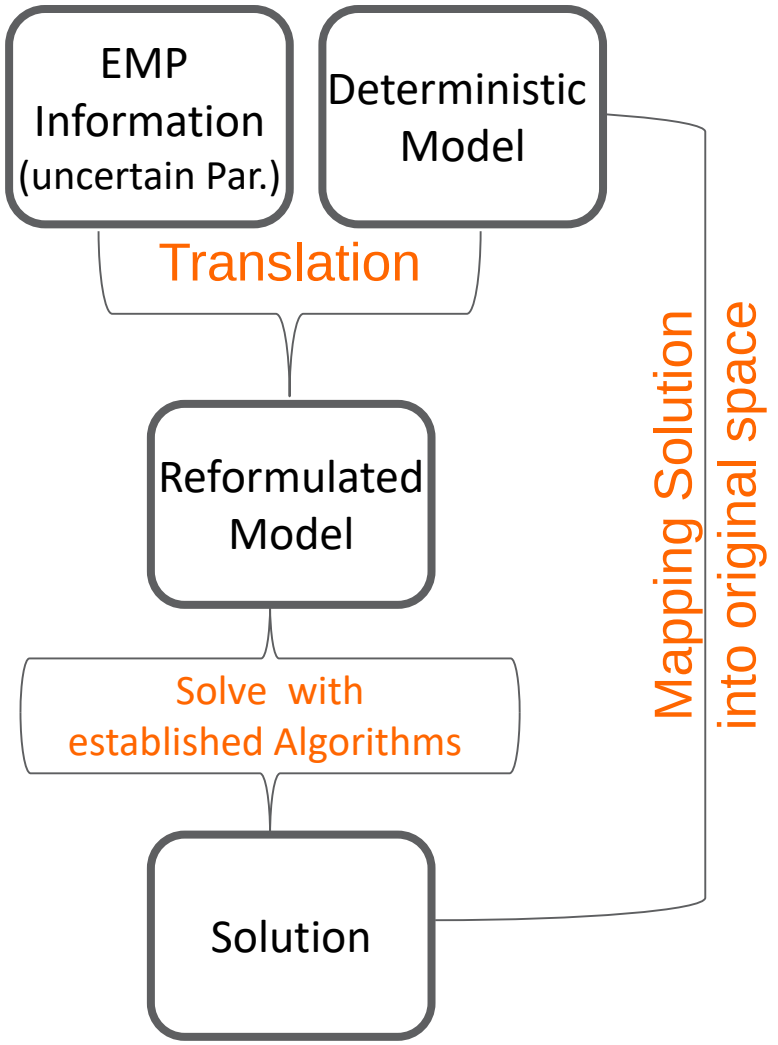
- gussgrid: https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_gussgrid.html
  - Asynchronous GUSS-solves in combined submission & collection loop. Keep number of submitted models <= #threads


LP
• Determine minimum transportation cost
• Result: city to city shipment volumes
MIP
• Discrete decisions
• E.g.: Ship at least 100 cases
MINLP
• Non-linearity
• E.g.: Decrease in unit cost with growing volumes
Scenarios
• SolveLink
• Grid Facility
• GUSS
SP
• Uncertainty
• E.g. Uncertain Demand

Stochastic Programming in GAMS

EMP/SP

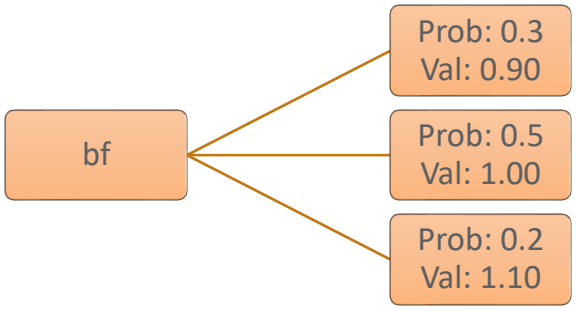
- Simple interface to add uncertainty to existing deterministic models
- (EMP) Keywords to describe uncertainty include: discrete and parametric random variables, stages, chance constraints, Value at Risk, …
- Available solution methods:
  - Automatic generation of **D**eterministic **E**quivalent (can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

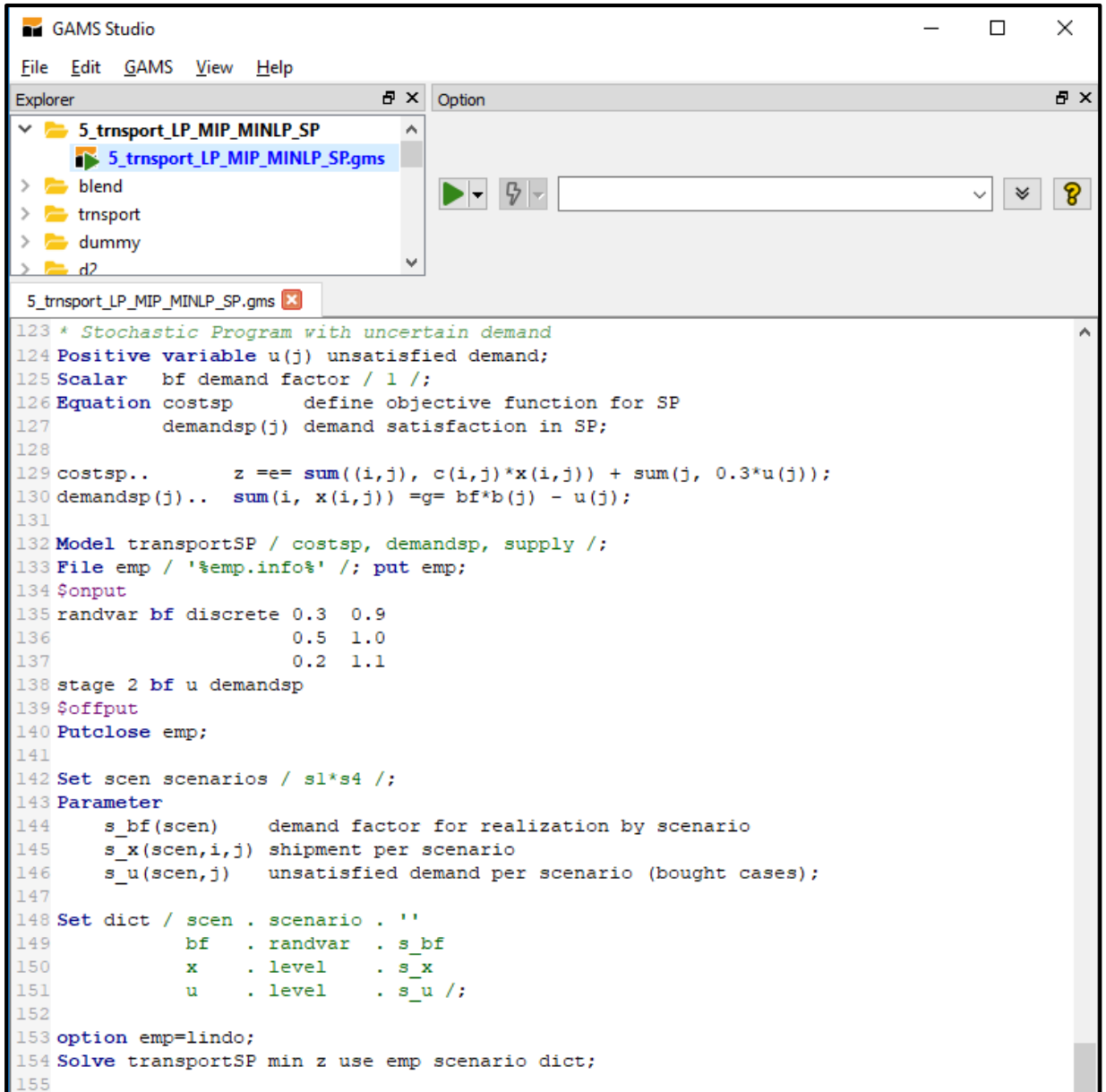


Decisions to make

- 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:

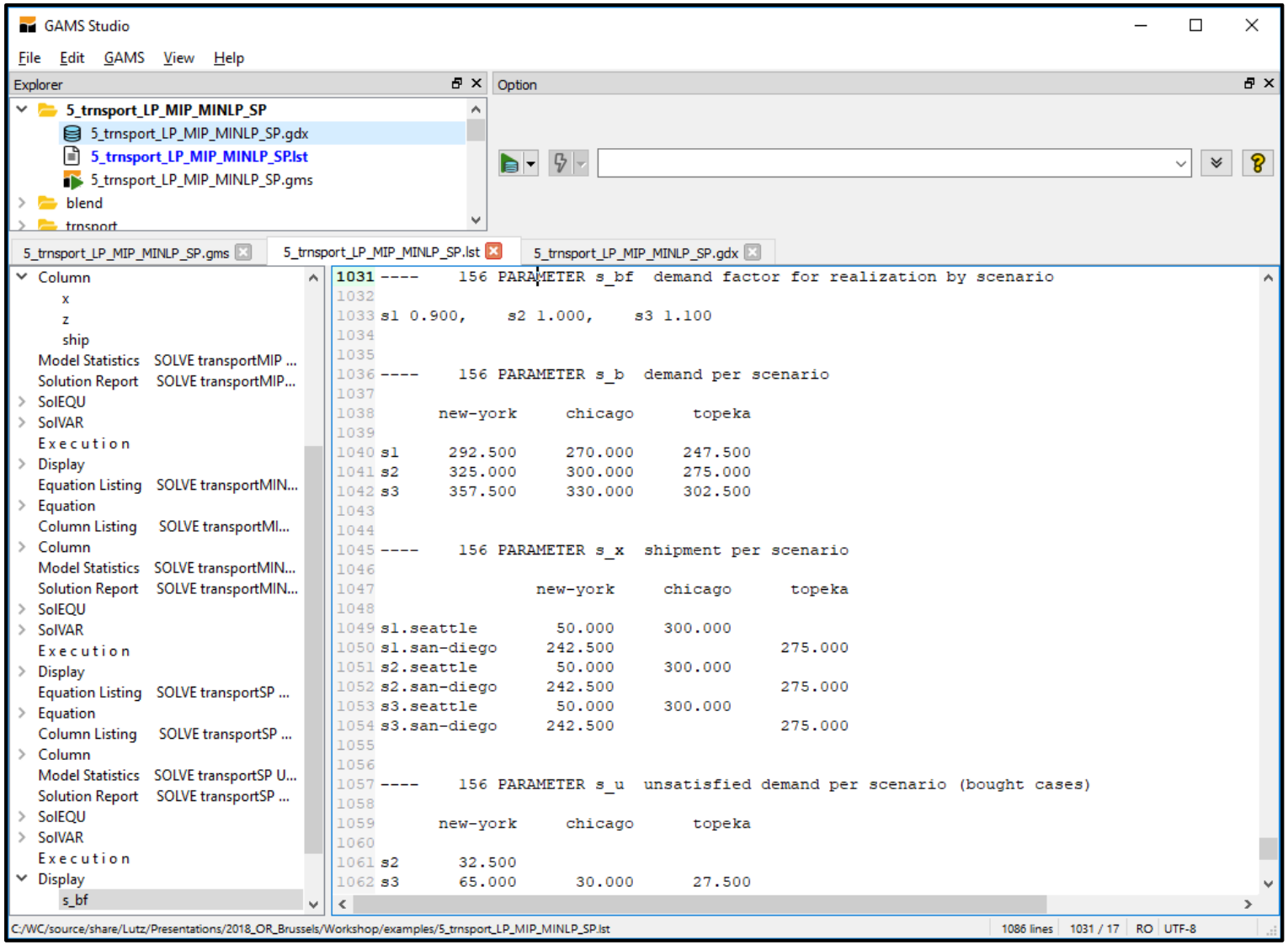
```
costsp ..   z   =e=   sum((i,j), c(i,j)*x(i,j))+ sum(j,0.3*u(j));
demandsp(j) .. sum(i, x(i,j))   =g=   bf*b(j) - u(j) ;
```

Uncertain Demand: GAMS Algebra



Hands-On

Uncertain Demand: Results



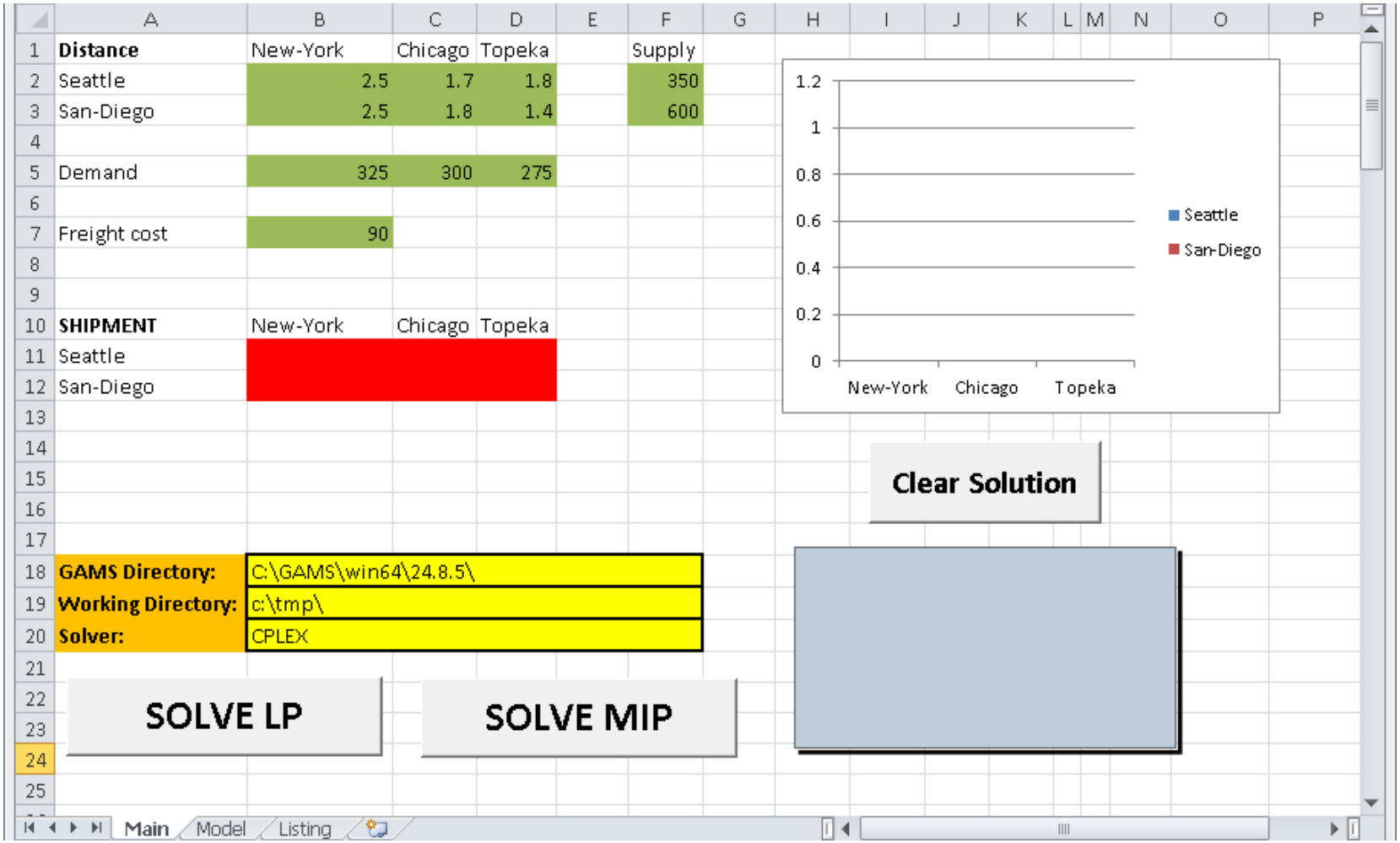
Stochastic Programming in GAMS

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

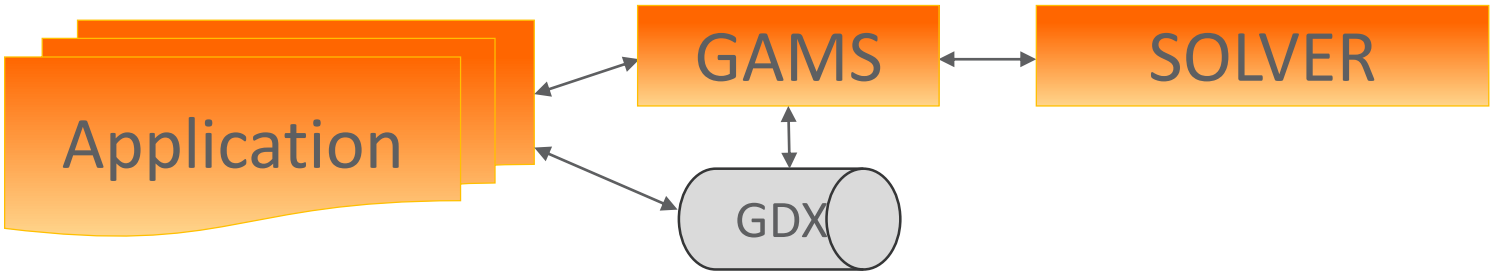
Excel and GAMS



	A	B	C	D	E	F
1	**Distance**	New-York	Chicago	Topeka		Supply
2	Seattle	2.5	1.7	1.8		350
3	San-Diego	2.5	1.8	1.4		600
4						
5	Demand	325	300	275		
6						
7	Freight cost	90				
8						
9						
10	**SHIPMENT**	New-York	Chicago	Topeka		
11	Seattle					
12	San-Diego					
13						
14						
15						
16						
17						
18	**GAMS Directory:**	C:\GAMS\win64\24.8.5\				
19	**Working Directory:**	c:\tmp\				
20	**Solver:**	CPLEX				
21						
22						
23						
24						
25						

Hands-On
TransXLS

Calling GAMS from your Application



Creating Input for GAMS Model

→ Data handling using **GDX** API

Callout to GAMS

→GAMS option settings using **Option** API

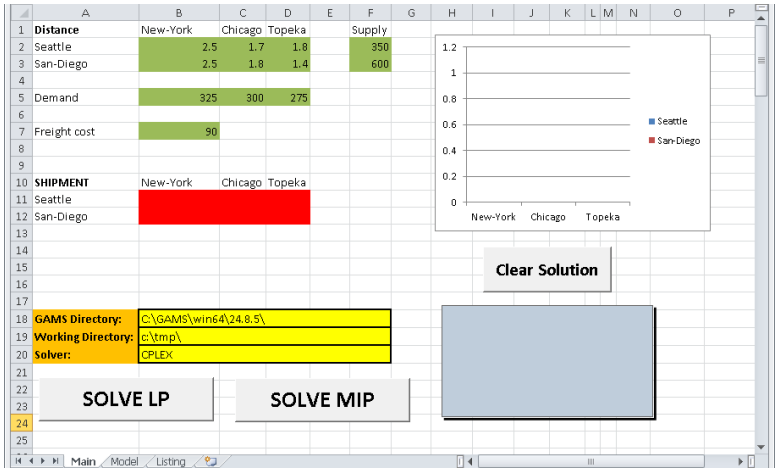
→Starting GAMS using **GAMS** API

Reading Solution from GAMS Model

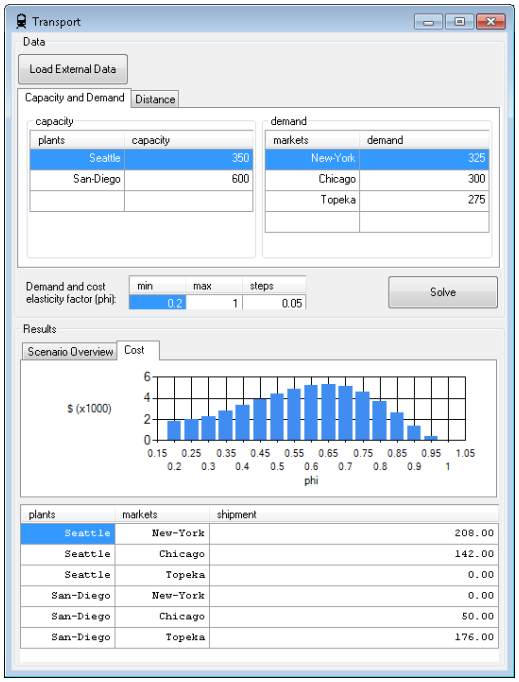
→ Data handling using **GDX** API

Low level APIs → Object Oriented API

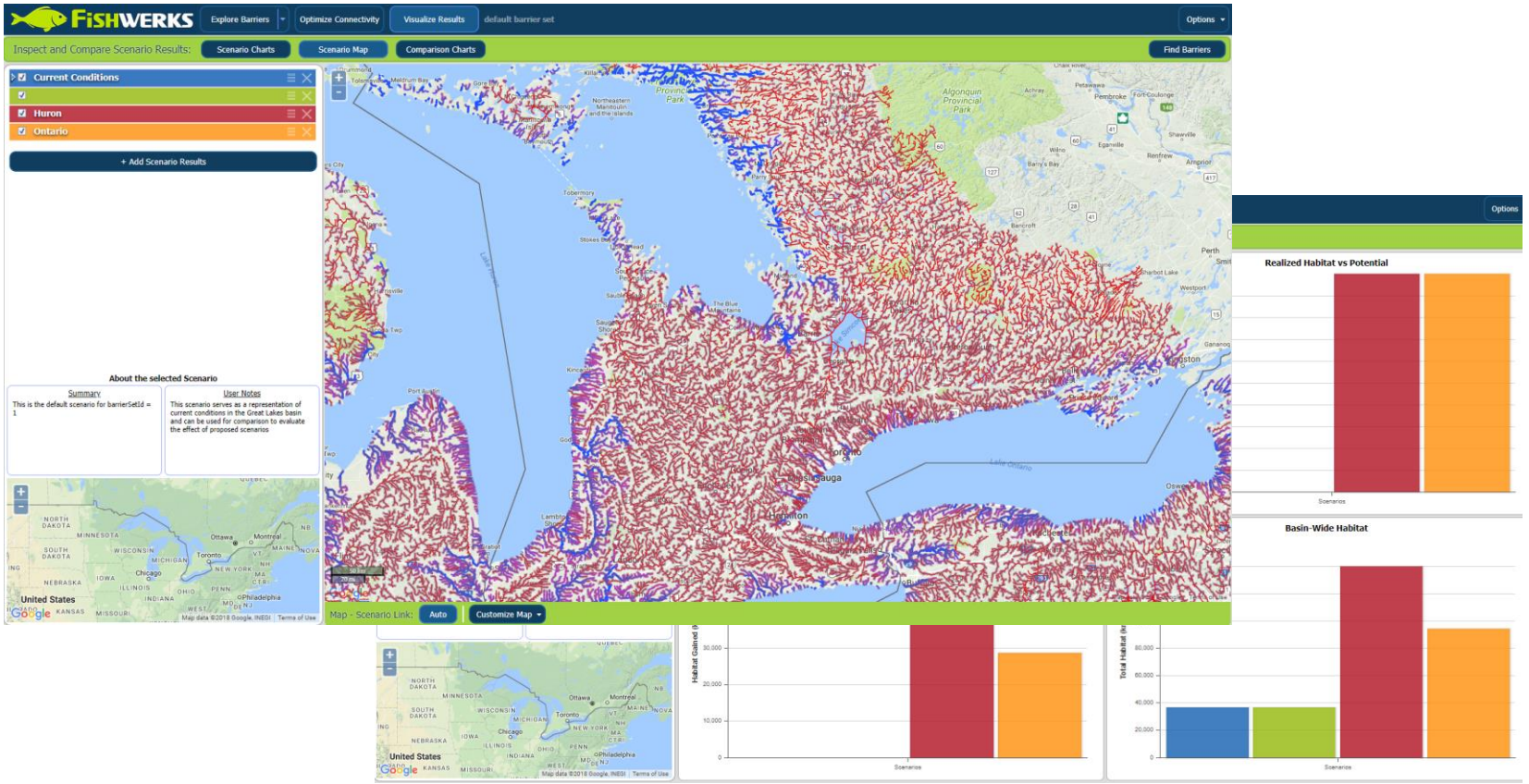
- Low level APIs
  - GDX, OPT, GAMSX, GMO, …
  - High performance and flexibility
  - Automatically generated imperative APIs for several languages:
    C, C++, C#, Delphi, Java, Python, VBA, …



- Object Oriented GAMS API
  - Additional layer on top of the low level APIs
  - Object Oriented
  - Written by hand to meet the specific requirements of different Object Oriented languages



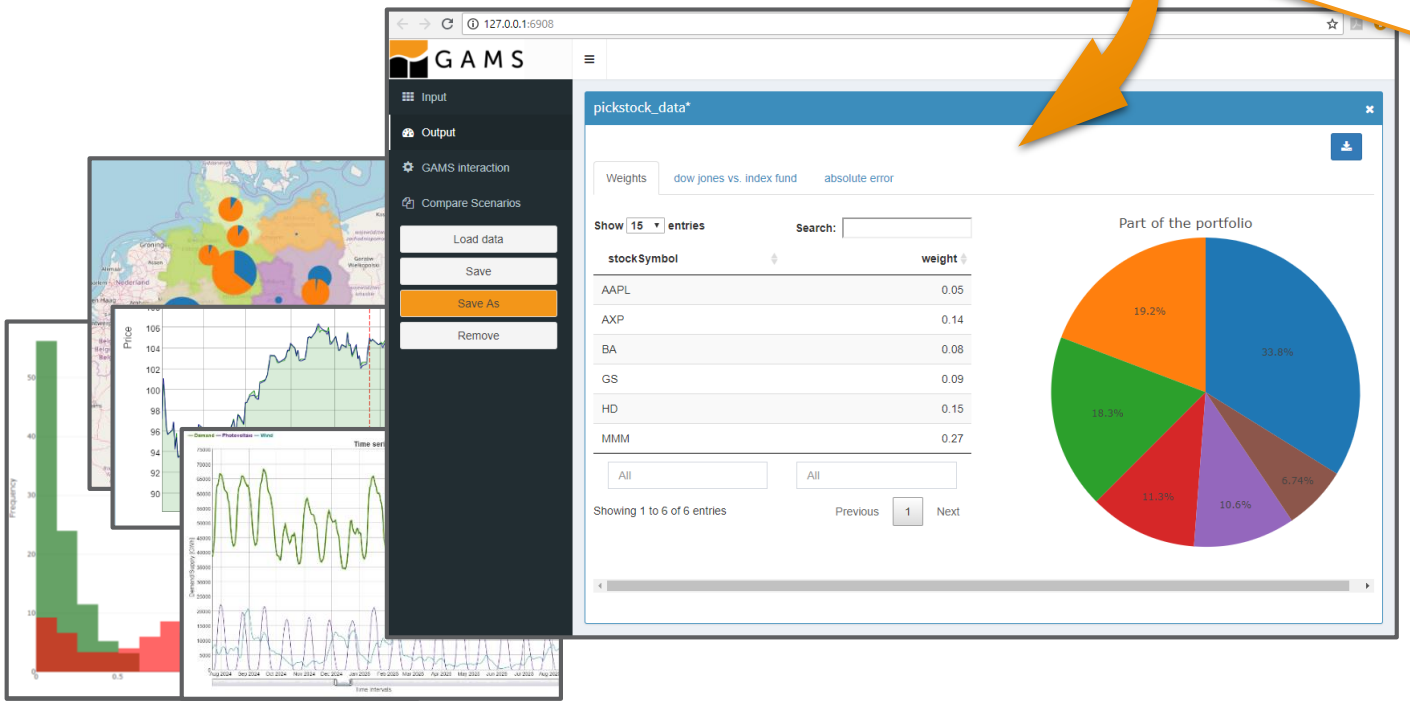
- GAMS comes with several OO APIs (Python, Java, C++, C#, …) to develop applications
  - → Programming required to build your applications



From GAMS Model to Visual Web User Interface

```
54 Equation
55     deffit(date)     'fit to Dow Jones index'
56     defpick(symbol) 'can only use stock if picked'
57     defnumstock      'few stocks allowed'
58     defobj           'absolute violation (L1 norm) from index';
59
60 deffit(ds)..   sum(s, price(ds,s)*w(s)) =e= index(ds) + slpos(ds) - slneg(ds);
61
62 defpick(s)..   w(s) =l= p(s);
63
64 defnumstock.. sum(s, p(s)) =l= maxstock;
65
66 defobj..       obj =e= sum(ds, slpos(ds) + slneg(ds));
```

Currently under Development



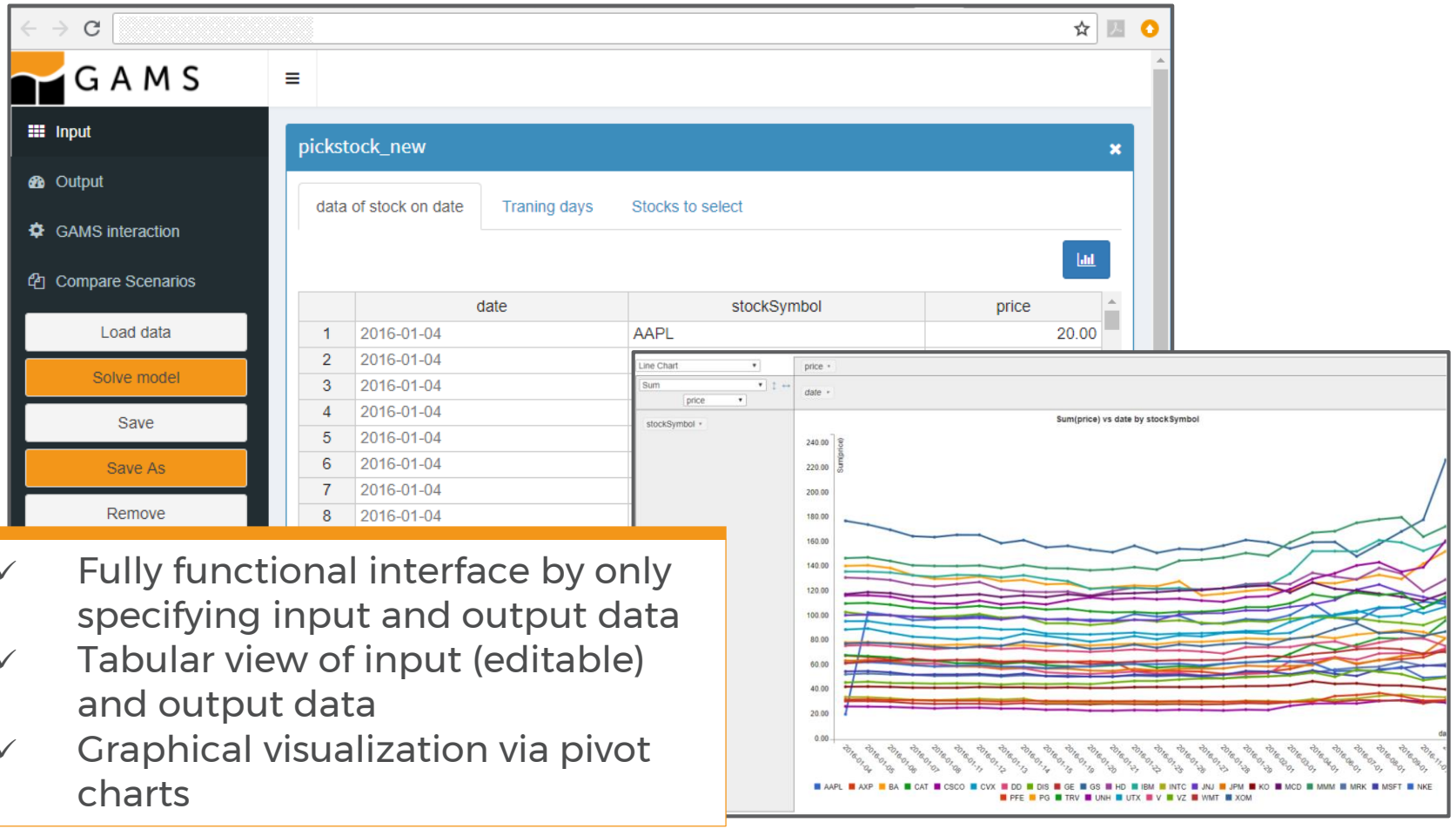
From GAMS Model to Visual Web User Interface

1

Basic setup:

✓ Annotating GAMS model (defining the input and output data to be displayed in the WebUI)

Basic Setup –
GAMS Model Annotations



- ✓ Fully functional interface by only specifying input and output data
- ✓ Tabular view of input (editable) and output data
- ✓ Graphical visualization via pivot charts

Basic Setup – GAMS Model Annotations

```
$onExternalInput
Set date   'date'
    symbol 'stockSymbol';

Parameter
   stockData(date,symbol,hrd)  'data of stock on date ### { "headers":{"date":{"readonly":true}} }';

Scalar
   maxstock      'maximum number of stocks to select ### { "slider":{"min":1, "max":"card(stockdata$
   tradingdays   'number of days for training        ### { "slider":{"min":1, "max":"card(stockdata$
$offExternalInput

$onExternalOutput
Set wHdr      'w header'               / 'weight'   /
    fHdr      'fund header'            / 'dow jones','index fund'  /
    errHdr    'stock symbol header'    / 'absolute error train', 'absolute error test' /;
Parameter
    partOfPortfolio(symbol,wHdr)       'what part of the portfolio'
    dowVSindex(date,fHdr)              'dow jones vs. index fund'
    abserror(date,errHdr)              'absolute error'
Singleton Set lastDayTraining(date)    'last date of training period ### vertical marker in chart' ;
$offExternalOutput

$if not exist webui.gms
$if set GMSWEBUI $abort Asked to do webui but can't find webui.gms. Set idir=path/to/webui
$batinclude webui
```

Basic Setup – GAMS Model Annotations

```
$onExternalInput
Set date '  date  '
    symbol 'stockSymbol';

Parameter
   stockData(date,symbol,hrd)  'data of stock on date ### { "headers":{"date":{"readonly":true}} }';

Scalar
   maxstock      'maximum number of stocks to select ### { "slider":{"min":1, "max":"card(stockdata$
                 of days for training        ### { "slider":{"min":1, "max":"card(stockdata$
$offExternalInput

$onExternalOutput
Set whdr      ' header'                  / 'weight'   /
    fHdr      'fund header'              / 'dow jones','index fund'  /
    errHdr    'stock symbol header'      / 'absolute error train', 'absolute error test' /;
Parameter
    partOfPortfolio(symbol,wHdr)         'what part of the portfolio'
    dowVSindex(date,fHdr)                'dow jones vs. index fund'
    abserror(date,errHdr)                'absolute error'
Singleton Set lastDayTraining(date)      'last date of training period ### vertical marker in chart' ;
$offExternalOutput

$if not exist webui.gms
$if set GMSWEBUI $abort Asked to do webui but can't find webui.gms. Set idir=path/to/webui
$batinclude webui
```

Basic Setup – GAMS Model Annotations

```
$onExternalInput
Set date   'date'
    symbol 'stockSymbol';

Parameter
   stockData(date,symbol,hrd)  'data of stock on date ### { "headers":{"date":{"readonly":true}} }';

Scalar
   maxstock      'maximum number of stocks to select ### { "slider":{"min":1, "max":"card(stockdata$
   trainingdays  'number of days for training        ### { "slider":{"min":1, "max":"card(stockdata$
$offExternalInput

$onExternalOutput
Set wHdr      'w header'               / 'weight'   /
    fHdr      'fund header'            / 'dow jones','index fund'  /
    errHdr    'stock symbol header'    / 'absolute error train', 'absolute error test' /;
Parameter
    partOfPortfolio(symbol,wHdr)       'what part of the portfolio'
    dowVSindex(date,fHdr)              'dow jones vs. index fund'
    abserror(date,errHdr)              'absolute error'
Singleton Set lastDayTraining(date)    'last date of training period ### vertical marker in chart' ;
$offExternalOutput

$if not exist webui.gms
$if set GMSWebUI Asked to do webui but can't find webui.gms. Set idir=path/to/webui
$batinclude webui
```

From GAMS Model to Visual Web User Interface

1

Basic setup:

✓ Annotating GAMS model (defining the input and output data to be displayed in the WebUI)

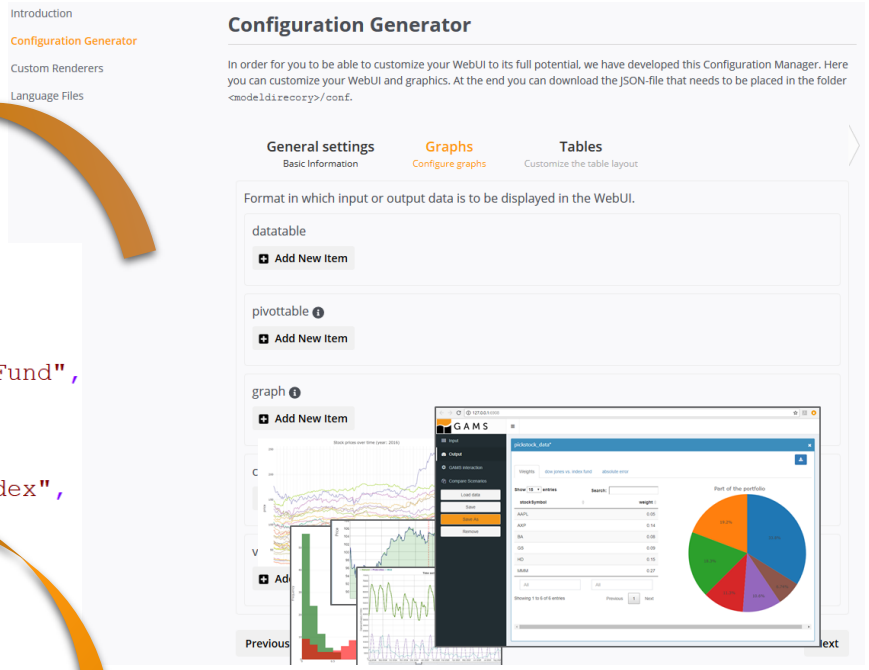
2

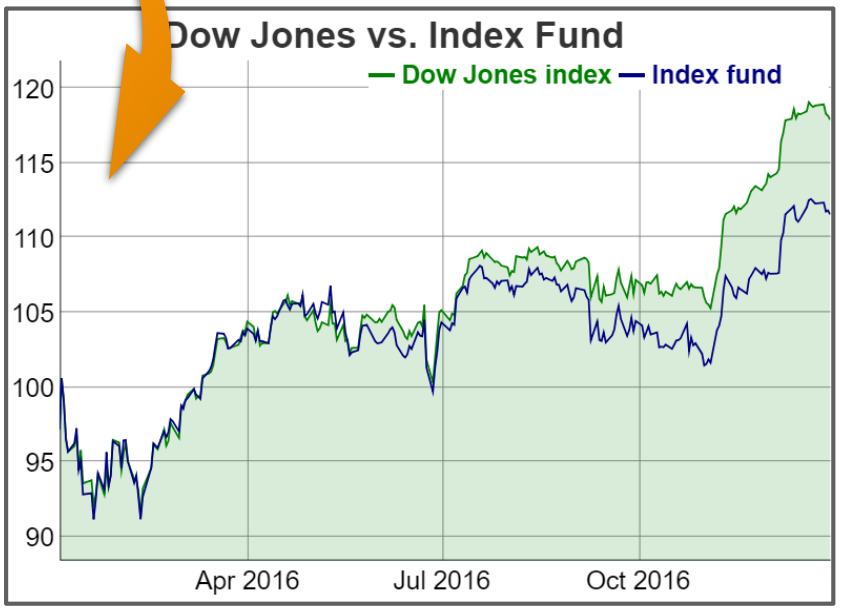
Advanced setup:

✓ Configuration of standard graphics and UI
✓ Sophisticated (custom) graphics (R API)
✓ Scenario management with internal database

Advanced Setup – Configuration

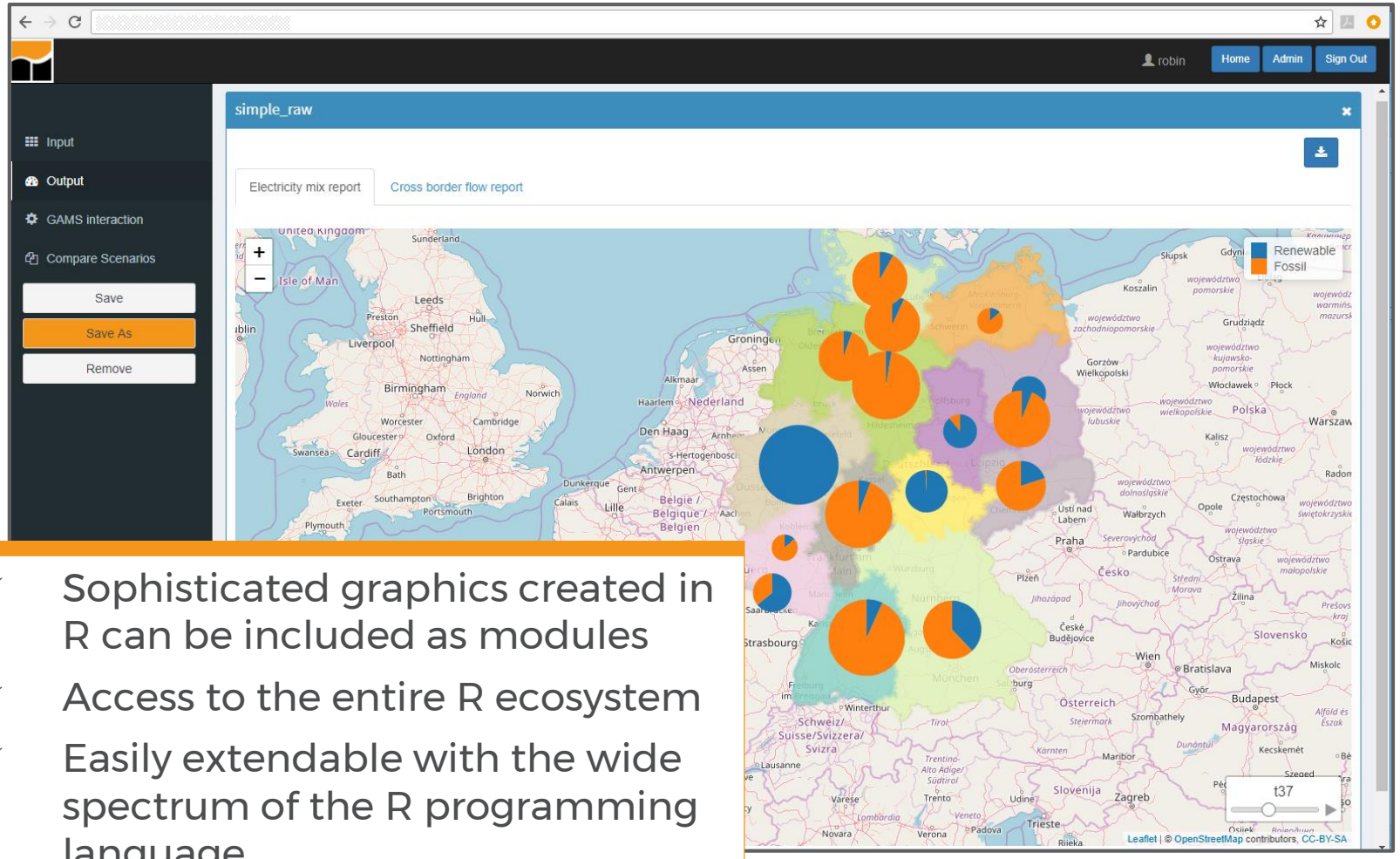
```
"dowvsindex"     : {
    "outType"                : "graph",
    "graph"                  : {
        "tool"                   : "dygraph",
        "title"                  : "Dow Jones vs. Index Fund",
        "xdata"                  : "date",
        "ydata"                  : {
            "dow jones"              : {
                "label"                  : "Dow Jones index",
                "fillGraph"              :
            },
            "index fund"
                "label"                  : "Index fund"
            }
        }
    }
}
```





- ✓ Configuration via JSON file (Configuration Generator)
- ✓ Access to a number of pre-implemented tools for graphical representation
- ✓ Focus on configuration, not programming

Advanced Setup – Sophisticated graphics



- ✓ Sophisticated graphics created in R can be included as modules
- ✓ Access to the entire R ecosystem
- ✓ Easily extendable with the wide spectrum of the R programming language

From GAMS Model to Visual Web User Interface

1

Initialization:

✓ Annotating GAMS model (defining the input and output data to be displayed in the WebUI)

2

Advanced setup:

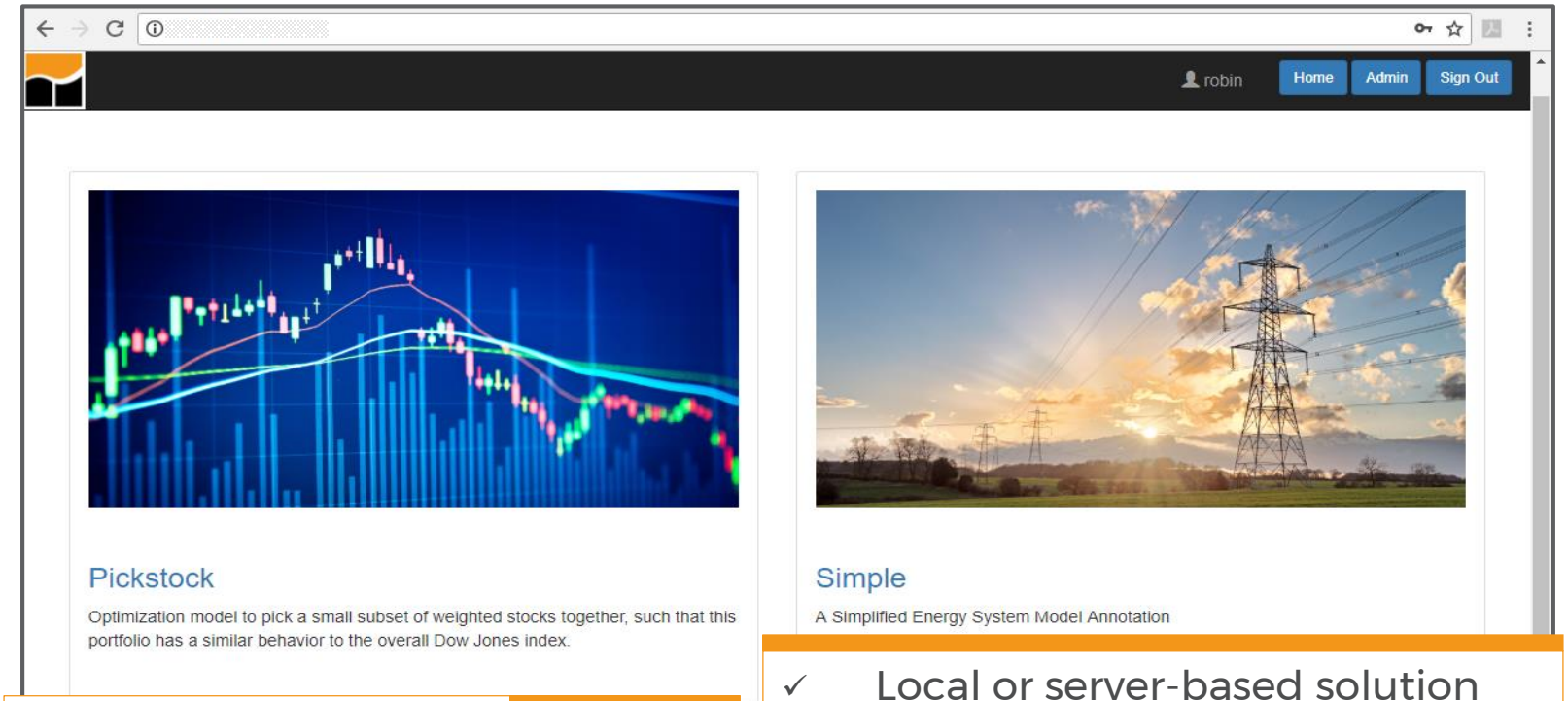
✓ Configuration of standard graphics and UI
✓ Sophisticated (custom) graphics (R API)
✓ Scenario management with internal database

3

Enterprise setup:

✓ User- and Application management

Enterprise Setup –
User and Application Management



Yaml config file

```
- name: simple
  display-name: Simple
  logo-url: file://localhost/home/ec2-user/simple.png
  description: A Simplified Energy System Model Annotation
  docker-cmd: ["R", "-e", "shiny::runApp('/root/GMSWebUI')"]
  docker-image: gams/app
  docker-network: "my-network"
  docker-env:
    SHINYPROXY_MODELNAME: simple
  groups: GAMS_team, guest
```

- ✓ Local or server-based solution
- ✓ User authentication (e.g. LDAP, OAuth 2.0, Google, GitHub, Facebook)
- ✓ Multi-Application support with docker-based technology

Hands-On

Enhanced Model Deployment in GAMS using R/Shiny

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

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

GAMS

Thank You