Pre Conference Workshop

Lutz Westermann lwestermann@gams.com

Clemens Westphal cwestpahl@gams.com

GAMS Software GmbH

GAMS Development Corporation www.gams.com

I. Stochastic Programming
II. Object Oriented GAMS API

I. Stochastic Programming

AML and Stochastic Programming (SP)

- Algebraic Modeling Languages/Systems good way to represent optimization problems
  - Algebra is a universal language
  - Hassle free use of optimization solvers
  - Simple connection to data sources (DB, Spreadsheets, …) and analytic engines (GIS, Charting, …)
- Large number of (deterministic) models in production
  - Opportunity for *seamless* introduction of new technology like Global Optimization, Stochastic Programming, …
  - AML potential framework for SP

2010

Simple Example

Simple Example: Newsboy (NB) Problem

- Data:
  - A newsboy faces a certain demand for newspapers
    d = 45
  - He can buy newspapers for fixed costs per unit
    c = 30
  - He can sell newspapers for a fixed price
    r = 60
  - For hold units he has to pay a disposal fee
    h = 10
  - He has to satisfy his customers demand or has to pay a penalty
    p = 5

- Decisions:
  - How many newspapers should he buy: ***X*** **45**
  - How many newspapers should he sell: ***S*** **45**

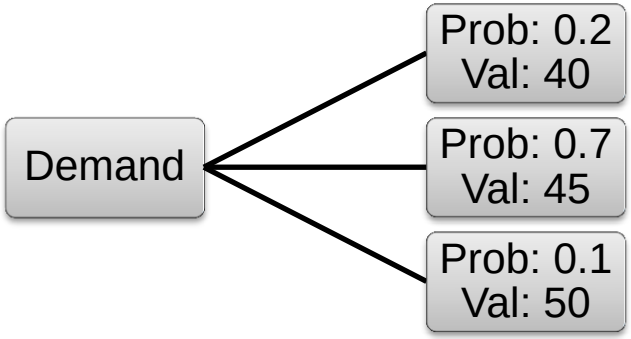
- Derived Outcomes:
  - How many newspapers need to be disposed: ***I*** **0**
  - How many customers are lost: ***L*** **0**

Simple NB Problem – GAMS Formulation

```
*           LostSales = demand - UnitsSold
lSales..    L =e= d - S;

*           Inventory = UnitsBought - UnitsSold
Inv..       I =e= X - S;

*           Profit, to be maximized
Profit..    Z =e= r*S - c*X - h*I - p*L;

Model nb / all /;

solve nb max z use lp;
```

NB Problem – Add Uncertainty

- Uncertain demand *d*



- Decisions to make:
  - How much newspaper should he buy “here and now” (without knowing the outcome of the uncertain demand)?
    → *First-stage decision*
  - How many customers are lost after the outcome becomes known?
    → *Second-stage* or *recourse decision*
  - Recourse decisions can be seen as
    - penalties for bad first-stage decisions
    - variables to keep the problem feasible

Stochastic NB Problem – GAMS Extension

```
* Make d uncertain
randvar d discrete  0.2 40
                    0.7 45
                    0.1 50

* Define non-default stages
stage 2 d I L S
stage 2 lSales Inv
```

➔ nbsimple.gms

New GAMS (EMP) Keywords

Excursus: EMP, what?

With new modeling and solution concepts do not:

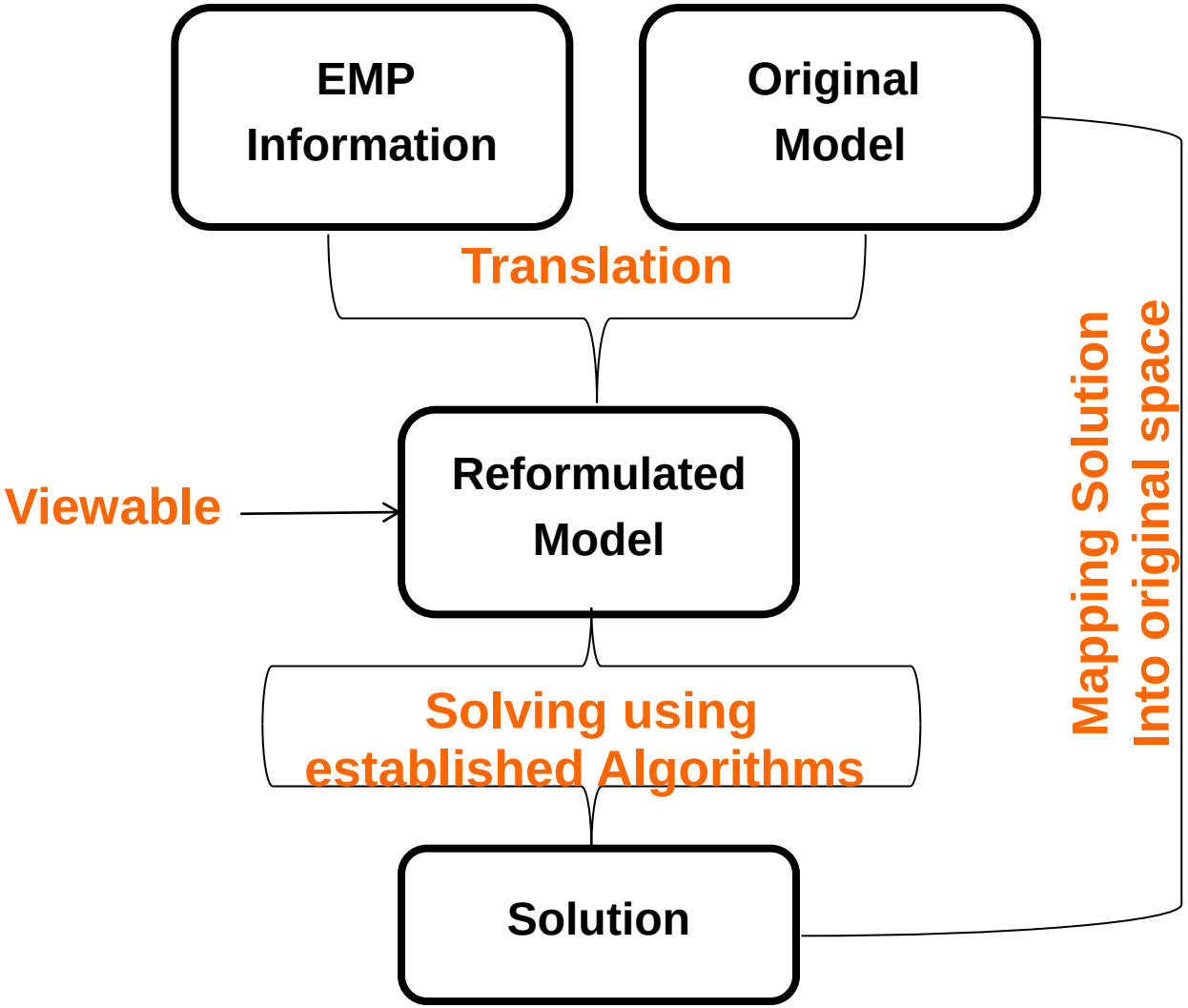
- overload existing GAMS notation right away !
- attempt to build new solvers right away !

But:

- Use existing language features to specify additional model features, structure, and semantics
- Express extended model in symbolic (source) form and apply existing modeling/solution technology
- Package new tools with the production system

→ **Extended Mathematical Programming (EMP)**

JAMS: a GAMS EMP Solver

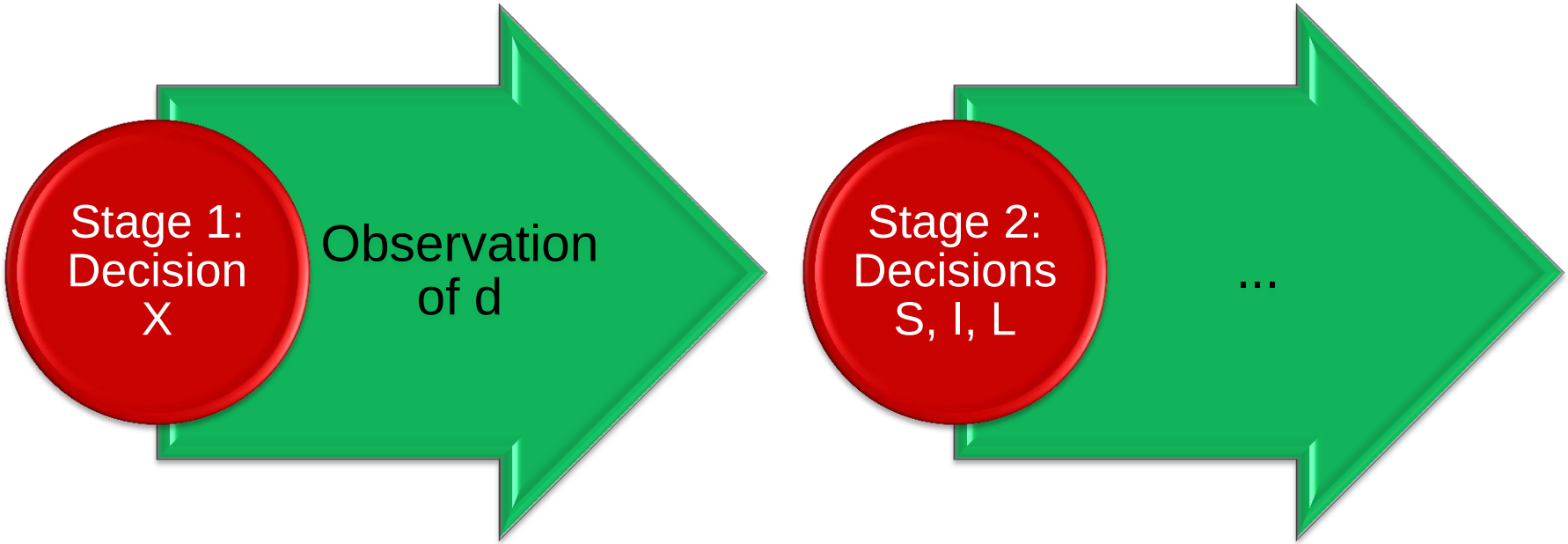


Output Extraction

- The expected value of the solution can be accessed via the regular `.L` and `.M` fields
- In addition, the following information can be stored in a parameter by scenario:
  - `level`: Levels of variables or equations
  - `marginal`: Marginals of variables or equations
  - `randvar`: Realization of a random variable
  - `opt`: Probability of each scenario
- This needs to be stored in a separate dictionary:

```
Set dict / scen.scenario .''
           x    .level     .s_x
           ''   .opt       .srep /;
```

Stages



Stages [stage]

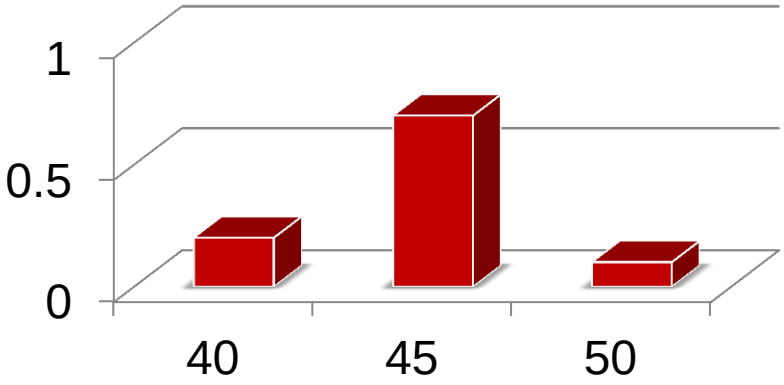
- Defines the stage of random variables (`rv`), equations (`equ`) and variables (`var`):

```
stage stageNo rv | equ | var {rv | equ | var}
```

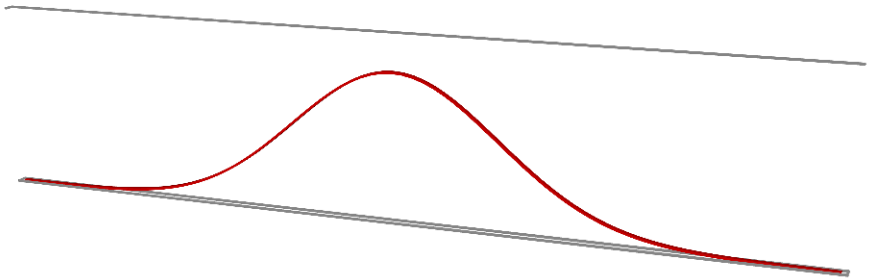
- `StageNo` defines the stage number
- The default `StageNo` for the objective variable and objective equation is the highest stage mentioned
- The default `StageNo` for all the other random variables, equations and variables not mentioned is 1

Random Variables

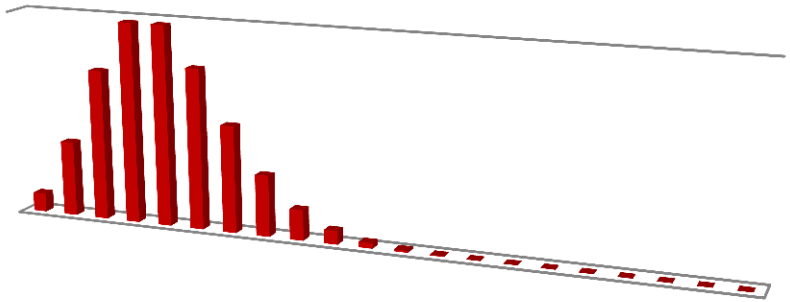
Discrete Distribution



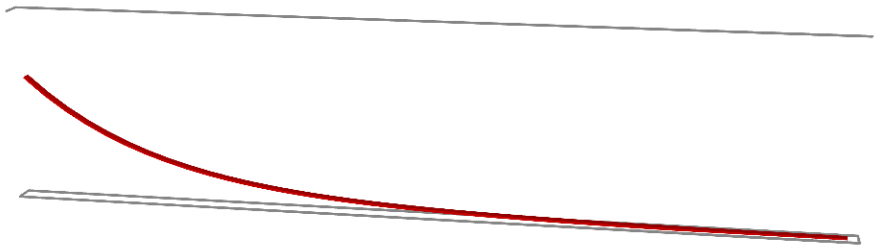
Normal Distribution



Poisson Distribution



Exponential Distribution



Random Variables (RV) [`randVar`]

- Defines both discrete and parametric random variables:

```
randVar rv discrete prob val {prob val}
```

- The distribution of discrete random variables is defined by pairs of the probability `prob` of an outcome and the corresponding realization `val`

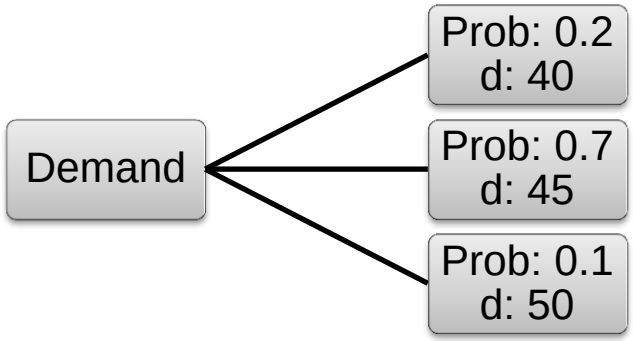
➔ **nbdiscindep.gms**

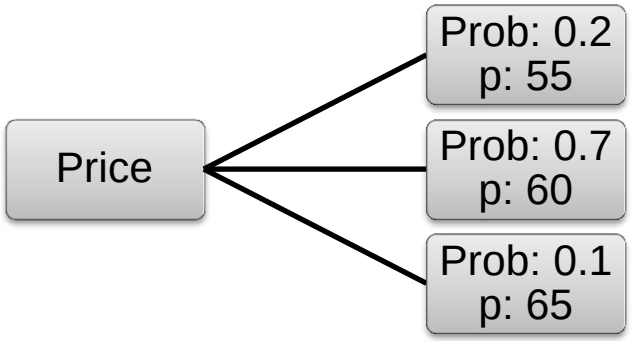
```
randVar rv distr par {par}
```

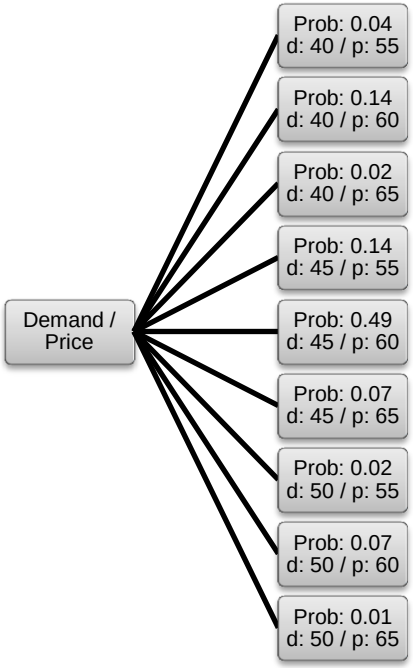
- The name of the parametric distribution is defined by `distr`, `par` defines a parameter of the distribution

➔ **nbcontindep.gms**

Joint Random Variables







vs.

Joint RVs [`jRandVar`]

- Defines discrete random variables and their joint distribution:

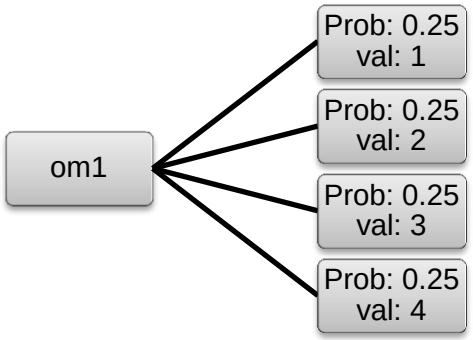
```
jRandVar rv rv {rv} prob val val {val}
                    {prob val val {val}}
```

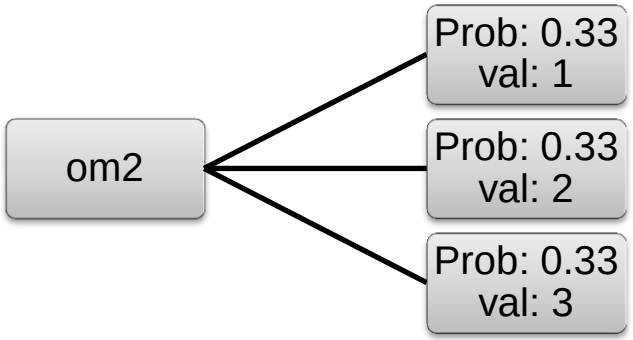
- At least two discrete random variables `rv` are defined and the outcome of those is coupled
- The probability of the outcomes is defined by `prob` and the corresponding realization for each random variable by `val`

➔ **nbdiscjoint.gms**

Chance Constraints

```
OBJ..  Z  =e= X1 + X2;
E1..   om1*X1 +   X2 =g=  7;
E2..   om2*X1 + 3*X2 =g= 12;
Model sc / all /;
solve sc min z use lp;
```






```
chance E1 0.6
chance E2 0.6
```

Chance Constraints

3 out of 4 must be true

[0.75 ≥ 0.6]

- `1*X1 + X2 =g= 7;`
- `2*X1 + X2 =g= 7;`
- `3*X1 + X2 =g= 7;`
- `4*X1 + X2 =g= 7;`

2 out of 3 must be true

[0.66 ≥ 0.6]

- `1*X1 + 3*X2 =g= 12;`
- `2*X1 + 3*X2 =g= 12;`
- `3*X1 + 3*X2 =g= 12;`

Chance Constraints [chance]

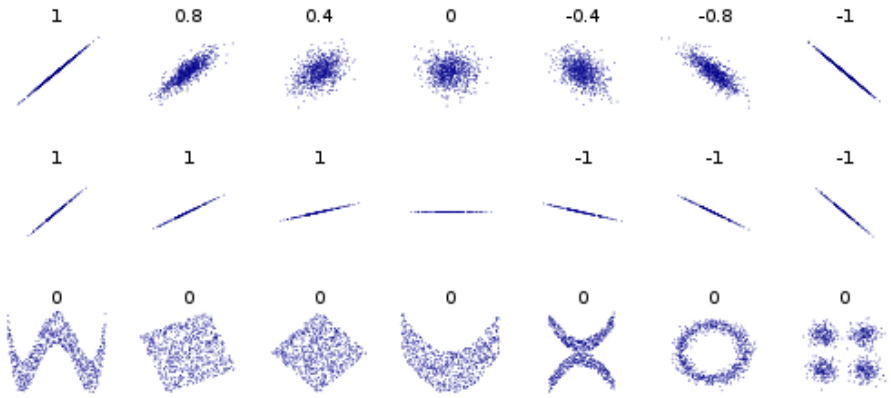
- Defines individual or joint chance constraints (CC):

```
chance equ {equ} [holds] minRatio [weight|varName]
```

- Individual CC: A single constraint `equ` has to hold for a certain ratio (0 ≤ `minRatio` ≤ 1) of the possible outcomes
- Joint CC: A set of constraints `equ` has to hold for a certain ratio (0 ≤ `minRatio` ≤ 1) of the possible outcomes
- If `weight` is defined, the violation of a CC gets penalized in the objective (weight violationRatio)
- If `varName` is defined the violation get multiplied by this existing variable

➔ **simplechance.gms**

Correlation between RVs [correlation]



Source: Wikipedia

- Defines a correlation between a pair of random variables:

```
correlation rv rv val
```

- `rv` is a random variable which needs to be specified using the `randvar` keyword and `val` defines the desired correlation (-1 ≤ `val` ≤ 1)

➔ **nbcontjoint.gms**

Adding Uncertainty to Transport

```
...
Model transport /all/ ;

file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
$onput
randvar b('new-york') normal 325 50
randvar b('chicago')  normal 300 50
randvar b('topeka')   normal 275 50
stage 2 b demand
$offput
putclose emp;

Set scen        scenarios / s1*s6 /;
Parameter
    s_b(scen,j)   demand realization by scenario
    s_x(scen,i,j) shipment per scenario
    s_s(scen) ;

Set dict / scen .scenario.''
           b    .randvar .s_b
           x    .level   .s_x /;

Solve transport using emp minimizing z scenario dict;
```

Summary

Available GAMS SP Solvers

	DE	DECIS	LINDO
chance	√		√
correlation			√
jrandvar	√	√	√
randvar (discrete)	√	√	√
randvar (parametric)			√

Conclusion

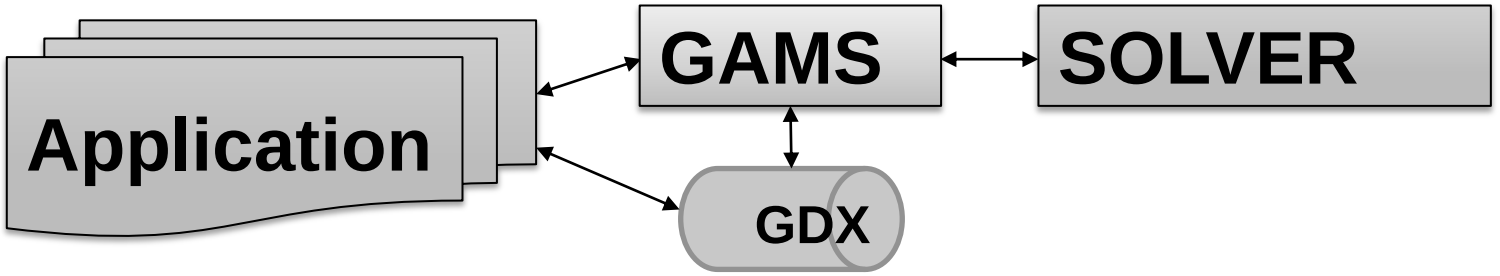
- Deterministic examples from all kind of application areas exist already (e.g. ~400 in the GAMS Model Library)
- Easy to add uncertainty to existing deterministic models, to
  - … either use specialized algorithms (DECIS, LINDO)
  - … or create Deterministic Equivalent and select from wide range of existing GAMS solver links (DE, free)
- New SP examples in the GAMS EMP Library
- More work to be done:
  - Scenario tree support
  - Sampling
  - …

II. Object Oriented GAMS API

Outline

- Introduction
- Development of a small GUI Application in C#

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, Delphi, Java, Python, C#, …)

- 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

Features of the object oriented API

- No modeling capability. Model is still written in GAMS
- Prepare input data and retrieve results in a convenient way → *GAMSDatabase*
- Control GAMS execution → *GAMSJob*
- Seamless integration of GAMS into other programming environments

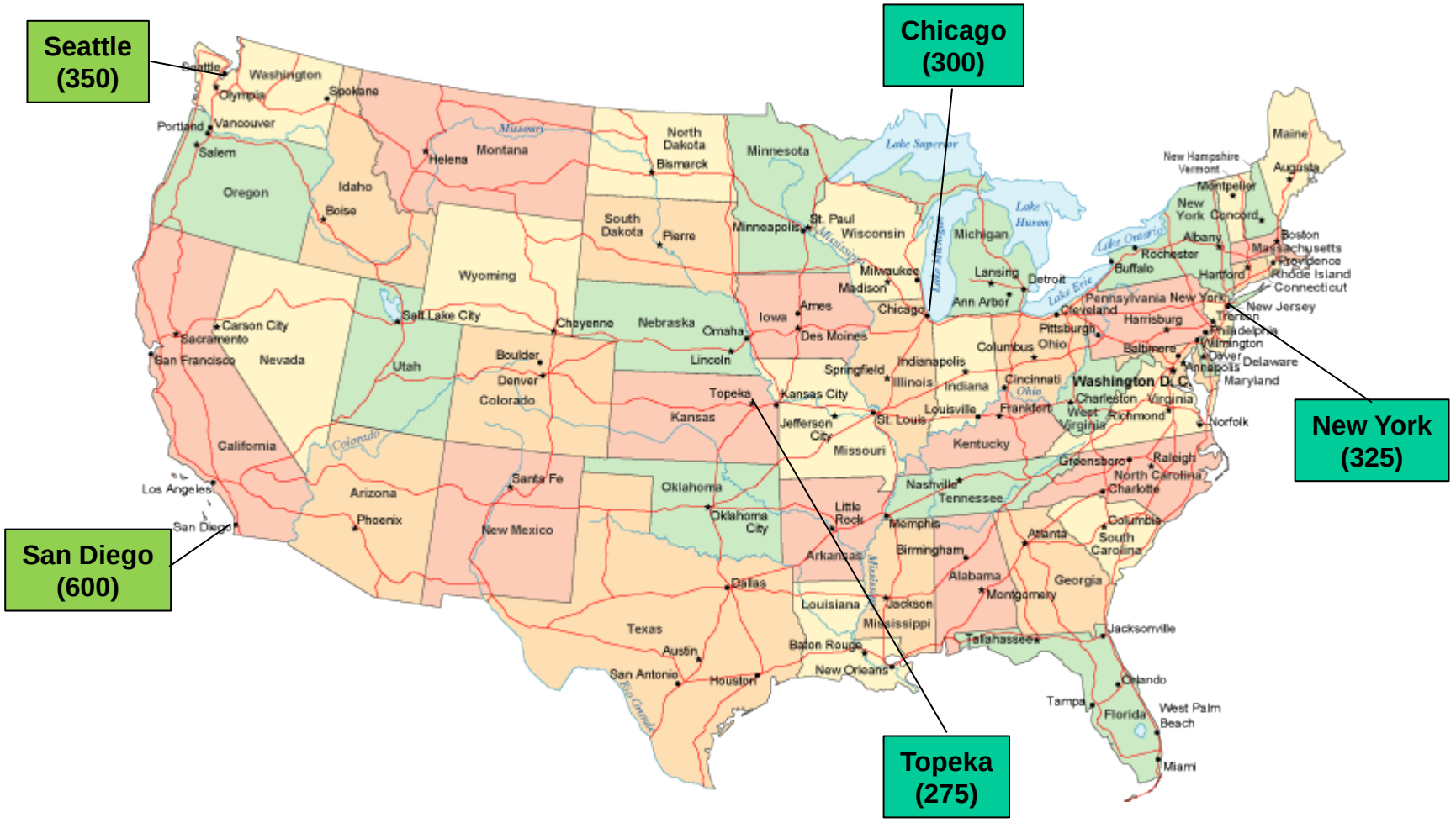
→ .NET API and several examples are part of the current GAMS release available at www.gams.com

Transport Application

- Scenario solves of the transportation problem (mutable scalar)

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

A Transportation Model



Summary

- Object Oriented API provides an additional abstraction layer of the low level GAMS APIs
- Powerful and convenient link to other programming languages
- .NET API is part of the current GAMS release available at *www.gams.com*. Many examples available:
  - Sequence of Transport examples
  - Cutstock, Warehouse, Benders Decomposition
- Python and Java under development.

Contacting GAMS

Europe

GAMS Software GmbH
Eupener Str. 135-137
50933 Cologne
Germany

Phone: +49 221 949 9170
Fax: +49 221 949 9171

info@gams.de

USA

GAMS Development Corp.
1217 Potomac Street, NW
Washington, DC 20007
USA

Phone: +1 202 342 0180
Fax: +1 202 342 0181

sales@gams.com
support@gams.com

http://www.gams.com