Pre-Conference Workshops

Michael Bussieck
Steve Dirkse
Fred Fiand
Lutz Westermann

Outline

- Part I: An Introduction to GAMS
- Part II: Stochastic programming in GAMS
- Part III: The GAMS Object-Oriented API's
- Part IV: Code embedding in GAMS

Stochastic Programming - Introduction

Stochastic programs are mathematical programs that involve **uncertain** data.

Motivation:

Real world problems frequently include some uncertain parameters. Often these uncertain parameters follow a probability distribution that is known or can be estimated.

Goal:

Find some policy that is feasible for all (or almost all) the possible data instances and that maximizes the expectation of some function of the decision variables and the random variables.

Example:

In a two-stage stochastic programming problem with recourse the decision maker has to make a decision now and then minimize the expected costs of the consequences of that decision.

Simple Example: Newsboy (NB) Problem

- Data:
  - A newsboy faces a certain demand for newspapers
    $d = 63$
  - He can buy newspapers for fixed costs per unit
    $c = 30$
  - He can sell newspapers for a fixed price
    $v = 60$
  - For leftovers he has to pay holding costs per unit
    $h = 10$
  - He has to satisfy his customers demand or has to pay a penalty
    $p = 5$

- Decisions:
  - How many newspapers should he buy: X 63
  - How many newspapers should he sell: S 63

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

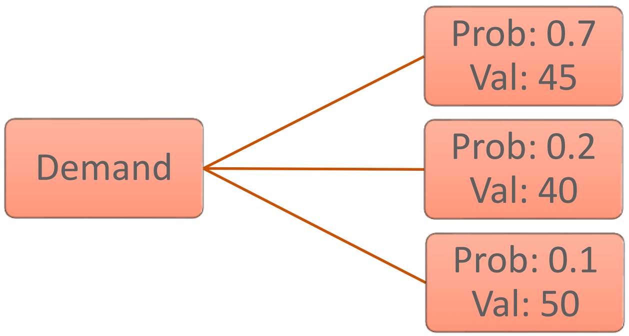
Simple NB Problem – GAMS Formulation

```
Variable Z Profit;
Positive Variables
         X Units bought
         I Inventory
         L Lost sales
         S Units sold;

Equations Row1, Row2, Profit;

* demand = UnitsSold + LostSales
Row1.. d =e= S + L;

* Inventory = UnitsBought - UnitsSold
Row2.. I =e= X - S;

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

Model nb / all /;


Solve nb max z use lp;
```

→ nbsimple.gms

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 newspapers are sold?
  - How many customers are lost after the outcome becomes known?
  - How many unsold newspapers go to the inventory?
    → ***Second-stage* or *recourse decisions***
  - Recourse decisions can be seen as
    - penalties for bad first-stage decisions
    - variables to keep the problem feasible

GAMS

Stochastic NB Problem – GAMS Extension

Idea:

Use deterministic model formulation plus some **annotation** to define uncertainty.

demand parameter

probability

Value of d

```
randvar d discrete 0.7  45
                   0.2  40
                   0.1  50
stage 2 I L S d
stage 2 Row1 Row2
```

Make demand d uncertain

Define (non-default) stage 2 variables and equations

Stochastic NB Problem – GAMS Extension

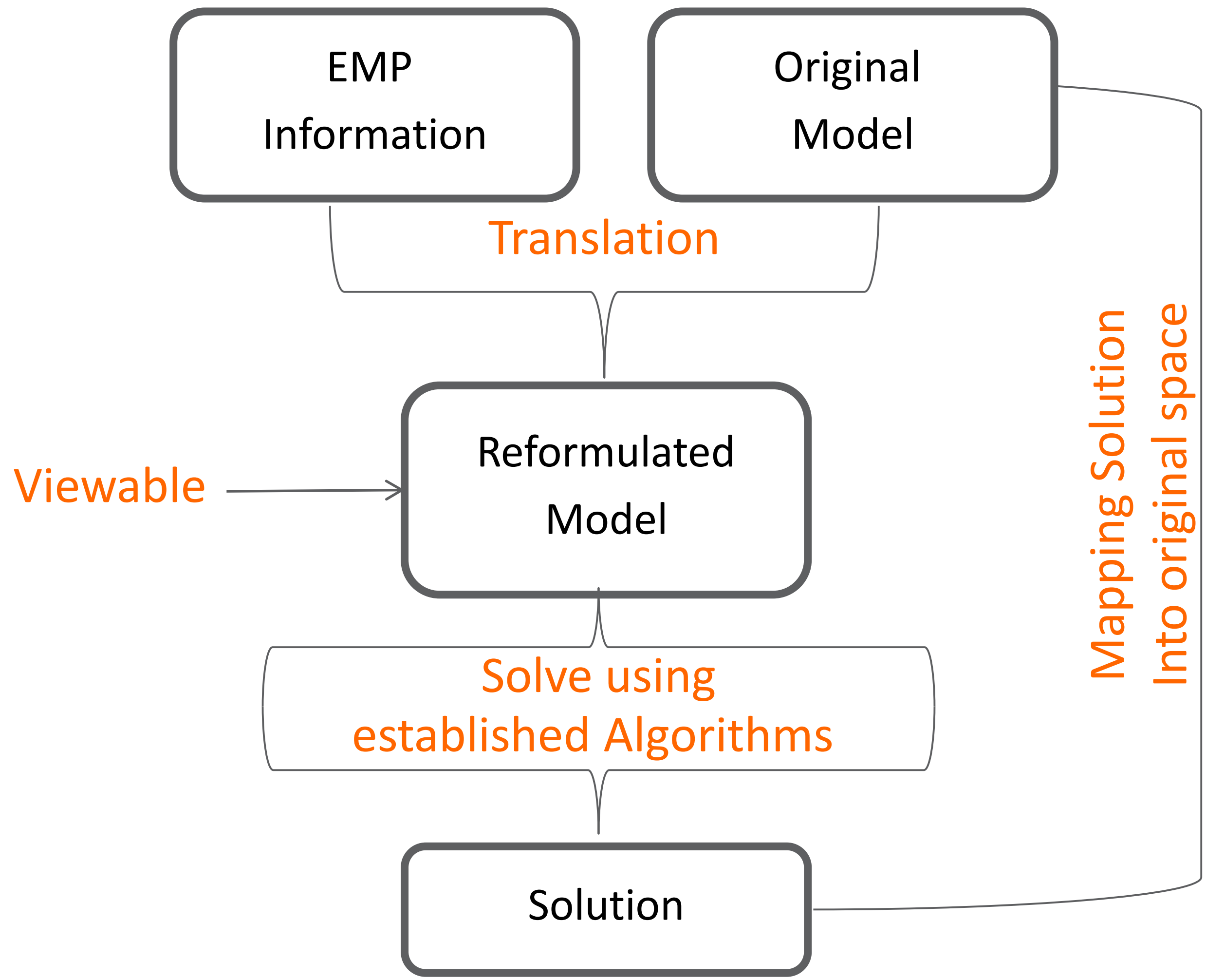
```
file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
$onput
randvar d discrete 0.7  45
                   0.2  40
                   0.1  50
stage 2 I L S d
stage 2 Row1 Row2
$offput
putclose emp;
```

Syntax to write an **EMP** info file on the fly, e.g. `[…]\225a\empinfo.dat`

EMP, what?
→ Excursus

The EMP Framework

EMP stands for **Extended Mathematical Programming**



Dictionary with output-handling information

- The expected value of the solution can be accessed via the regular `.L` (and `.M`) fields
- Additional information can be stored in a parameter by scenario, e.g.:
  - `level`: Levels 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 scen          Scenarios / s1*s3 /;
Parameter
    s_d(scen)     Demand realization by scenario
    s_x(scen)     Units bought by scenario
    s_s(scen)     Units sold by scenario
    s_o(scen,*)   scenario probability / #scen.prob 0 /;

Set dict / scen .scenario.''
           d    .randvar .s_d
           s    .level   .s_s
           x    .level   .s_x
           ''   .opt     .s_o /;

solve nb max z use emp scenario dict;
```

3 parts of a GAMS EMP stochastic model

1. The deterministic *core* model

2. EMP annotations in EMP info file

3. The *dictionary* with output-handling information

→ nbsimple.gms

Extensions to the Simple NB Problem

- Multiple stages:

→ nbdiscindep.gms

`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

- Several probability distributions for random variables:
  - Discrete distributions:

    `randVar rv discrete prob val {prob val}`
  - Continuous distributions: normal, binomial, exponential, …

    `randVar rv distr par {par}`

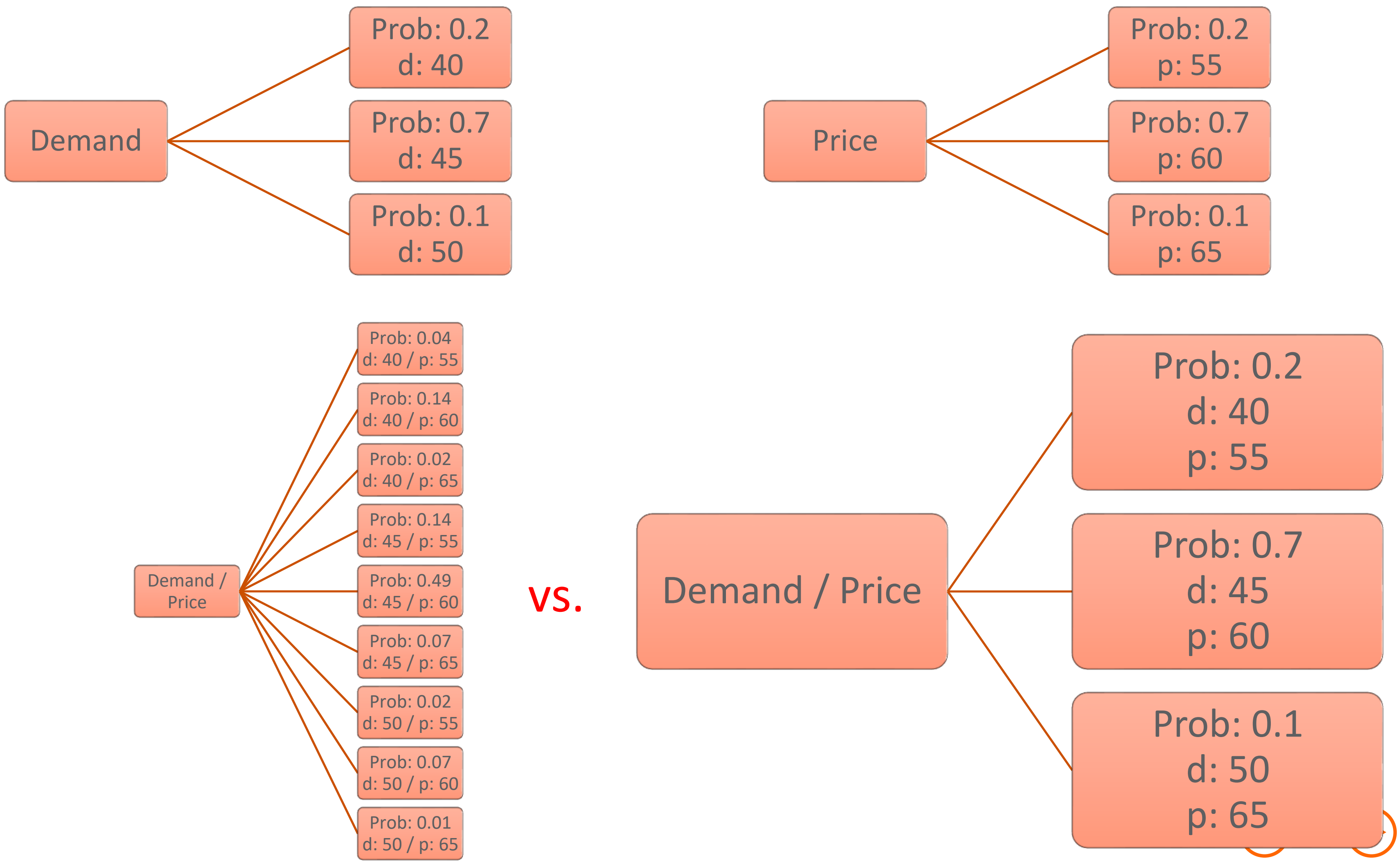
    `sample rv {rv} sampleSize`

→ nbdiscindep.gms

→ nbcontindep.gms

- Joint Random variables:

Independent vs. Joint Random Variables



Extensions to the Simple NB Problem

- Multiple stages:

→ nbdiscindep.gms

```
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

- Several probability distributions for random variables:
  - Discrete distributions:

→ nbdiscindep.gms

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

  - Continuous distributions: normal, binomial, exponential, …

→ nbcontindep.gms

```
randVar rv distr par {par}
sample rv {rv} sampleSize
```

- Joint Random variables:

→ nbdiscjoint.gms

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

Chance Constraints with EMP

```
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 with EMP

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

SP in GAMS - Summary & Outlook

- The Extended Mathematical Programming (EMP) framework can be 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 (e.g. solvers Lindo, DECIS) or create Deterministic Equivalent (free solver DE)
- Besides the expected value, EMP also supports optimization of other risk measures (e.g. VaR)
- GAMS/Scenred2 interfaces GAMS with the well-known scenario reduction software Scenred (https://www.gams.com/latest/docs/T_SCENRED2.html)
- More information: https://www.gams.com/latest/docs/UG_EMP_SP.html

Thank You!
Meet us at the GAMS booth!

Extended Example: Newsboy (NB) Problem

- Data:
  - A newsboy faces a certain demand for newspapers
    d = 63
  - He can buy newspapers for fixed costs per unit
    c = 30
  - He can sell newspapers for a fixed price
    v = 60
  - For leftovers he has to pay holding costs per unit
    h = 10
  - He has to satisfy his customers demand or has to pay a penalty
    p = 5
  - **He can return units for a refund (stage 3)**
    r = 9
- **Stage 1: Decisions:**
  - How many newspapers should he buy: ***X***
- ***Stage 2: Decisions & Derived Outcomes***
  - How many newspapers should he sell: ***S***
  - How many newspapers go to his inventory: ***I***
  - How many customers are lost: ***L***
- ***Stage 3: Decisions & Derived Outcomes***
  - *How many units returned for refund:* ***Y***
  - *How many units kept for holding cost h again* ***E***

Stages [stage]

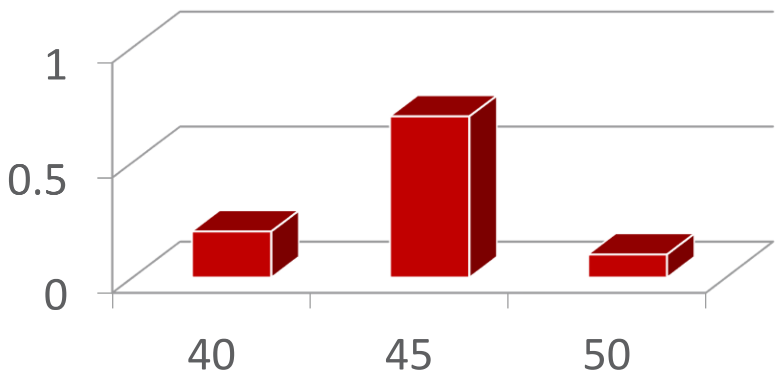
- Defines the stage of random variables (`rv`), equations (`equ`) and variables (`var`):

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

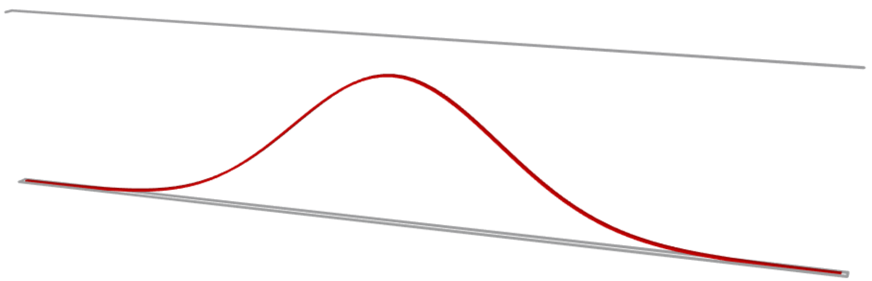
- `StageNo` defines the stage number
- The default `StageNo` for the objective variable and objective equation is the highest stage mentioned
- The default `StageNo` for all the other random variables, equations and variables not mentioned is 1

Random Variables

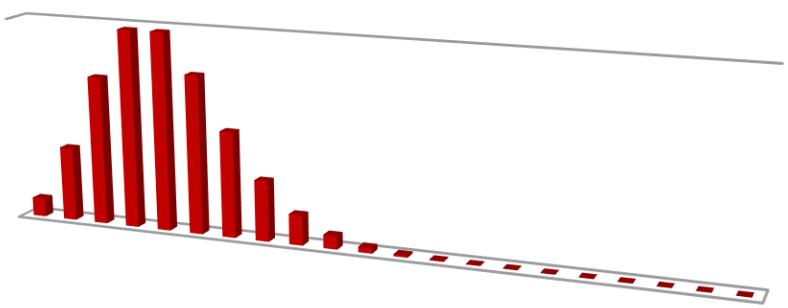
Discrete Distribution



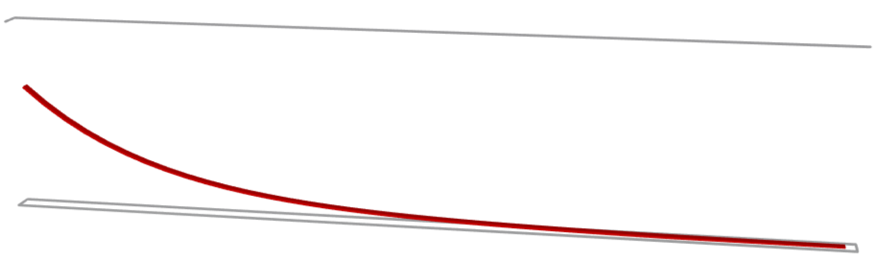
Normal Distribution



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.

For parametric distributions a sample can be created.

→ nbcontindep.gms

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

Correlation between RVs [`correlation`]

- 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