# GAMS/SCENRED

Nicole Gröwe-Kuska

Humboldt-University Berlin, Institute of Mathematics

(Vattenfall Europe Trading Gmbh)

Content:

1. GAMS/SCENRED
   - **1.1** Organization of GAMS/SCENRED programs
   - **1.2** Representation of scenarios (tree-structured or not) in GAMS models
   - **1.3** Input parameters
   - **1.4** Output parameters
   - **1.5** Diagnostic checks of the input parameters
   - **1.6** SCENRED Options
   - **1.7** Steps to build or modify a model for use with GAMS/SCENRED
2. Applying GAMS/SCENRED to the WINVEST Example

GAMS Workshop "Advanced risk management", September 2, 2003

# 1 GAMS/SCENRED

- GAMS/SCENRED introduced to GAMS Distribution 20.6 (May 2002)
- SCENRED is a collection of C++ routines for the optimal reduction of scenarios or scenario trees
- GAMS/SCENRED provides the link from GAMS programs to the scenario reduction algorithms. The reduced problems can then be solved by a deterministic optimization algorithm provided by GAMS.
- SCENRED contains three reduction algorithms:
  - FAST BACKWARD method
  - Mix of FAST BACKWARD/FORWARD methods
  - Mix of FAST BACKWARD/BACKWARD methods

  Automatic selection (best expected performance w.r.t. running time)

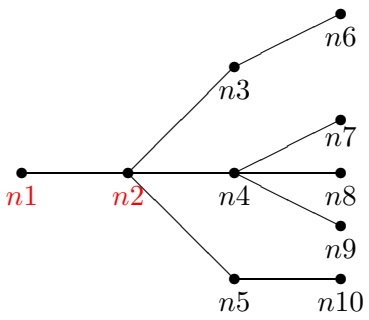
## 1.1 Organization of GAMS/SCENRED programs

| Component | Contents |
|---|---|
| 1. DATA | ○ set & parameter declarations and definitions<br>○ `$libinclude scenred.gms`<br>○ assignments<br>○ displays |
| 2. SCENRED CALL | ○ export the initial scenarios from GAMS to SCENRED<br>○ execute SCENRED<br>○ import the reduced scenarios from SCENRED to GAMS |
| 3. MODEL | ○ variable declaration<br>○ equation declarations<br>○ equation definitions (using sets from reduced tree)<br>○ model definition & solution |

## 1.2 Representation of scenarios (tree-structured or not) in GAMS models

- by using a set of nodes
- by using an ancestor matrix (a dynamic set indicating the ancestor-successor or father-child relationship between any pair of nodes)
- by assigning a probability (at least to the leaf) nodes, and
- by assigning a given number of random values to any node. The random values can be in separate parameters.

Example for an ancestor matrix:




| Node/Ancestor | n1 | n2 | n3 | n4 | n5 | n6 | n7 | n8 | n9 | n10 |
|---|---|---|---|---|---|---|---|---|---|---|
| n1 | T | F | F | F | F | F | F | F | F | F |
| n2 | T | T | F | F | F | F | F | F | F | F |
| n3 | F | T | T | F | F | F | F | F | F | F |
| n4 | F | T | F | T | F | F | F | F | F | F |
| n5 | F | T | F | F | T | F | F | F | F | F |
| n6 | F | F | T | F | F | T | F | F | F | F |
| n7 | F | F | F | T | F | F | T | F | F | F |
| n8 | F | F | F | T | F | F | F | T | F | F |
| n9 | F | F | F | T | F | F | F | F | T | F |
| n10 | F | F | F | F | T | F | F | F | F | T |

### 1.3 Input parameters

- initial scenario set
- GAMS parameter `ScenRedParms` contains
  1. statistics describing the initial scenarios (number of nodes, number of leaves, number of time steps, number of random values assigned to a node)
  2. specifies the desired reduction (number of preserved scenarios or reduction percentage)
  3. optional parameters (choice of reduction method, limit on running time)

### 1.4 Output parameters

- Revised node probabilities for nodes in reduced scenario set
- Revised ancestor matrix for reduced scenario set
- A `ScenRedReport` GAMS parameter, containing information about the SCENRED run, including:
  1. absolute and relative distance of the scenario sets
  2. running time
  3. number of nodes in the reduced scenario set

### 1.5 Diagnostic checks of the input parameters

- Consistency of the desired input parameters with the contents of the SCENRED input file (number of nodes, number of leaves, number of time steps, number of random values assigned to a node)
- Range check of desired input parameters and options
- Check of scenario and node probabilities.
    1. The scenario probabilities are rescaled if they do not sum up to 1.
    2. If probabilities of inner nodes are missing or wrong, assign node probabilities that are consistent with the scenario probabilities.
- Check of the ancestor matrix (check the orientation of the graph, check if the graph contains a cycle, check if the graph contains incomplete forests or scenarios)

### 1.6 SCENRED Options

- Names of input, option and output files
- Select a reduction algorithm
- Limit the running time
- Additional information about the stochastic model (number of stages, where do random data enter the model?)

## 1.7 Steps to build or modify a model for use with GAMS/SCENRED

- Analyse the GAMS program of the stochastic programming model. Since the initial scenarios and a number of input parameters have to be passed to SCENRED, one must identify the corresponding components of the GAMS model and create or calculate them if they do not already exist.
- Reformulate the GAMS program.
  Check if the model can handle varying scenario or node probabilities, and whether the equations are defined in terms of a (possibly reduced) tree. If the model doesn't already contain a scenario tree, one should be added. If it does, it is a simple task to rewrite the equation definitions (and possibly other statements too) in terms of a subset of the original nodes or tree.
- Add the statements for passing the initial set of scenarios to SCENRED, for the execution of SCENRED and for the import of the reduced scenarios from SCENRED.

# 2 Applying GAMS/SCENRED to the WINVEST Example

## 2.1 Analyse the GAMS program of the WINVEST Example

The **structure of the initial scenario tree** is given by the dynamic set `ancestor`. The GAMS parameters `yieldN(i,n)`, `cash_yieldN(n)` and `liabN(n)` contain the **random parameters** of the stochastic program.
The SCENRED parameter `SCENREDPARMS` contains the following information:

- Number of scenarios: `card(Cscen)`=8 scenarios
- Number of nodes: `card(Cnodes)`=12 nodes
- Number of time steps: `card(Ctime)`
- Number of random values assigned to node n: `card(i)`+2 (`yieldN(i,n)`, `cash_yieldN(n)` and `liabN(n)`)

```
$libinclude scenred.gms
ScenRedParms('num_leaves') =  card(Cscen);
ScenRedParms('num_nodes') = card(n);
ScenRedParms('num_time_steps') = card(Ctime);
ScenRedParms('num_random') = card(i)+2;
```

There is no GAMS parameter for the **node probabilities**. Since the variable `ExpWealth` represents the final expected wealth of our portfolio, the scenario probabilites can be found in the objective `TOT_FINAL`: the scenarios have equal probabilities `1/card(Cscen)`.

The node probabilities of the initial scenario tree will be contained in the parameter prob. We will define the probabilities of the leaves; the probabilities of the remaining nodes are computed during the SCENRED run:

```
parameter prob(n) "node probability of the initial tree";
prob(leaf(n))=1/card(Cscen);
```

The parameters red_num_leaves and red_percentage control the tree output from ScenRed; at least one of them is required:

```
ScenRedParms('red_num_leaves') = card(CScen)/2;
ScenRedParms('red_percentage') = 0.5;
```

We add dynamic sets for the nodes in the reduced subtree and their probabilities.

```
set sn(n) "subset of nodes in reduced subtree";
parameter sprob(n) "node probability in reduced tree";
```

It remains to set up the SCENRED option file:

```
* set up the scenred options file
file opts / 'scenred.opt' /;
putclose opts  'log_file    SRLogWinStoch.txt'
             / 'input_gdx   SRInWinStoch.gdx'
             / 'output_gdx  SROutWinStoch.gdx';
```

### 2.2 Reformulate the GAMS program to handle varying node probabilities

Since WINSTOCH cannot handle varying scenario or node probabilities, we add the missing components to reformulate the GAMS model.
The **decision variables** associated with the nodes in the reduced scenario tree are x(i,sn(n)), buy(sn(n),i), sell(sn(n),i), final(sn(n)) and cash(sn(n)). Hence, the decision variables are already declared.
We now reformulate the constraints CONST1, CONST2, CASHC and TOT_FINAL to depend on sn.

```
equations SRCNST1(n), SRCNST2(n0, n, i), SRCASHC(n0,n),
          SRTOT_FINAL;
* -- Initial holdings sum to 100:
  SRCNST1(root(sn(n0))) ..
          sum(i, x(n0, i)) + cash(n0) =E= 100;
* -- At each second-stage (non-root) node, we "receive"
     the initial (root) investment times its yield,
     and can sell if desired. Can also buy.
*    This defines the holdings "out of" the node:
  SRCNST2(ancestor(n0,sn),i) ..
     x(n0,i) * yieldN(i,sn) +  # Coming into node sn(n)
     buy(sn, i)
                  =E=
     sell(sn, i) +
     x(sn, i) $ (not leaf(sn));# Going out of node sn,
                               # except leaves.
```

```
  SRCASHC(ancestor(n0,sn)) .. # Keep track of cash position
     cash(n0) * cash_yieldN(sn) +
     sum(i, sell(sn, i)) * (1-transcost)
                 =E=
     sum(i, buy(sn, i))  +
     liabN(sn) $ liabs +
     cash(sn); #cash out of leaf nodes are final positions
  SRTOT_FINAL ..          # Total final position:
     ExpWealth =E= sum(sn$leaf(sn), sprob(sn)*cash(sn));
* Can't buy at a leaf node, can't sell at the root node:
  buy.fx(leaf(sn), i)  = 0;
  sell.fx(root(sn), i) = 0;
model SRWinStoch /SRCNST1,SRCNST2,SRCASHC,SRTOT_FINAL/;
solve SRWinStoch maximizing ExpWealth using lp;
```

### 2.3 Statements for passing the initial set of scenarios to SCENRED, for the execution of SCENRED and for the import of the reduced scenarios from SCENRED

```
scalar    rc;
execute_unload 'SRInWinStoch.gdx',  ScenRedParms, n,
          ancestor, prob, yieldN,cash_yieldN,liabN ;
execute 'scenred scenred.opt %system.redirlog%';
     rc = errorlevel;
     abort$rc "Return code from scenred is nonzero:",rc;
execute_load   'SROutWinStoch.gdx', ScenRedReport,
       sprob=red_prob;
```

Nodes with nonzero probability form the node set of the reduced scenario tree

```
sn(n)=sprob(n);
```