SCENRED

Contents

1 Introduction . . . 651
2 Scenario Reduction Algorithms . . . 651
3 Using GAMS/SCENRED . . . 652
4 The SCENRED Input File . . . 653
5 SCENRED Options and the Option File . . . 655
6 The SCENRED Output File . . . 656
7 Diagnostic Check of Scenario Trees . . . 657
8 SCENRED Errors and Error Numbers . . . 657
9 SCENRED Warnings . . . 658

Release Notes

- May, 2002: Level 001 (GAMS Distribution 20.6)
  - GAMS/SCENRED introduced.

1 Introduction

Stochastic programs with recourse employing a discrete distribution of the random parameters become a deterministic programming problem. They can be solved by an appropriate optimization algorithm, ignoring the stochastic nature of (some or all) parameters.

SCENRED is a tool for the reduction of scenarios modeling the random data processes. The scenario reduction algorithms provided by SCENRED determine a scenario subset (of prescribed cardinality or accuracy) and assign optimal probabilities to the preserved scenarios. The reduced problem is then solved by a deterministic optimization algorithm provided by GAMS.

2 Scenario Reduction Algorithms

Many solution methods for stochastic programs employ discrete approximations of the uncertain data processes by a set of scenarios (i.e., possible outcomes of the uncertain parameters) with corresponding probabilities.

For most practical problems the optimization problem that contains all possible scenarios (the so-called deterministic equivalent program) is too large. Due to computational complexity and to time limitations this program is often approximated by a model involving a (much) smaller number of scenarios.

The reduction algorithms developed in [1,2] determine a subset of the initial scenario set and assign new probabilities to the preserved scenarios. All deleted scenarios have probability zero.

SCENRED contains three reduction algorithms: The Fast Backward method, a mix of Fast Backward/Forward methods and a mix of Fast Backward/Backward methods. In general, the computational performance (accuracy, running time) of the methods differ. For huge scenario trees the Fast Backward method has the best expected performance with respect to running time. The results of the Forward and Backward methods are more accurate, but at the expense of higher computing time. The Forward method is the best algorithm when comparing accuracy, but it can only be recommended if the number of preserved scenarios is small (strong reduction). The combined methods improve the result of the Fast Backward method if the Forward or Backward method, respectively, can be completed within the running time limit. If no reduction method is selected, the method with the best expected performance with respect to running time is chosen.

The reduction algorithms exploit a certain probability distance of the original and the reduced probability measure. The probability distance trades off scenario probabilities and distances of scenario values. Therefore, deletion will occur if scenarios are close or have small probabilities.

The reduction concept is general and universal. No requirements on the stochastic data processes (e.g. the dependency or correlation structure of the scenarios, the scenario probabilities or the dimension of the process) or on the structure of the scenarios (e.g. tree-structured or not) are imposed. The reduction algorithms can be tailored to the stochastic model if the user provides additional information (How many decision stages are involved? Where do the random parameters enter the model – in objective and/or right hand sides and/or technology matrices?) The information is used to choose the probability distances (cf. Remark 1 in [1]).

References (download: `www-iam.mathematik.hu-berlin.de/~romisch/RecPubl.html`)

I J. Dupačová, N. Gröwe-Kuska, W. Römisch: Scenario reduction in stochastic programming: An approach using probability metrics. Revised version to appear in Mathematical Programming.

II H. Heitsch, W. Römisch: Scenario reduction algorithms in stochastic programming. Preprint 01-8, Institut für Mathematik, Humboldt-Universität zu Berlin, 2001.

3 Using GAMS/SCENRED

The reduction algorithms require additional data preparation and reformulation of the GAMS program for the stochastic programming model.

GAMS offers great flexibility with respect to the organization of data specification, model definition and solve statements. The most common way to organize GAMS/SCENRED programs is shown below. Since the initial scenarios and a number of input parameters have to be passed to SCENRED, the corresponding components of the GAMS program have to be defined before the SCENRED call. The reduced scenarios have to be defined before the equations of the (reduced) stochastic programming model are used in a solve statement. Therefore the SCENRED call can be placed anywhere between the definitions of the GAMS parameters and the solve statement of the reduced stochastic programming model.

When building or modifying a model for use with GAMS/SCENRED the following steps should be taken:

- 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 (see Section 4), 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.

A reduction of the initial scenarios makes sense only if we are able to generate that part of the model that corresponds to the preserved scenarios (i.e. the reduced subtree). This is done by declaring a subset of the nodes in the original tree.

The parameters and equations are declared over the original node set, but are defined over only the subtree. This will be illustrated by an example later in the section.

Further, one should verify that the model can handle changing probabilities. Many practical models involve scenarios with equal probabilities. This property will not be maintained by the probabilities in the reduced subtree.

ORGANIZATION OF GAMS/SCENRED PROGRAMS

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

Prior to calling SCENRED, you should include the declaration of the SCENRED input and output parameters and the definition of the sets they are indexed by from the GAMS include library:

```
$libinclude scenred.gms
```

Once you have created all the inputs to SCENRED and assigned values to them, you are ready to write the SCENRED GDX data input file, write the SCENRED options file, call SCENRED, and read the reduced tree data from the SCENRED GDX data output file (see Sections 4,5,6). Assuming your model is formulated to use a subtree of the original, you can now continue with the solve and any subsequent reporting.

SCENRED is executed by issuing the statement

```
execute 'scenred optfilename';
```

where `optfilename` is the name of the SCENRED option file.

As an example, consider the `srkandw` model in the GAMS model library, and the `kand` model upon which it is based (get these from the modlib now!). To produce `srkandw` from `kand`, we first reformulate the original to allow for solution over a reduced tree. To do this, we introduce a subset of the node set: `set sn(n) 'nodes in reduced tree';` For convenience and clarity, we introduce a second subset at the same time, the set of leaf nodes: `set leaf(n) 'leaf nodes in original tree';` as well as some code to compute this set based on the existing time-node mapping. We also declare a new parameter, the probabilities for the reduced tree: `parameter sprob(n) 'node probability in reduced tree';` Once these are declared, we can quickly edit the equation *definitions* so that they run only over the reduced subtree: we simply substitute the reduced probabilities `sprob` for the original `prob`, and the reduced node set `sn` for the original node set `n`. Note that the *declaration* of the equations does not change.

This example illustrates one other change that may be required: the stochastic data must be in parameters having the node set as their last index. This is not the case in the `kand` model, so we simply reversed the indices in the `dem` parameter to meet the requirement in `srkandw`. It is also possible to create a transposed copy of the original data and pass that the SCENRED if the original data cannot be changed conveniently.

4 The SCENRED Input File

The SCENRED input file contains the initial scenarios and their stochastic parameter data, as well as statistics describing this input and (possibly) options to control the SCENRED run. This input file has a special binary format; it is a GDX (GAMS Data Exchange) file. The name of the SCENRED input file is assigned in the option file (see Section 5).

The scalar inputs to SCENRED are collected in the one-dimensional parameter `ScenRedParms`, the first parameter stored in the SCENRED input file. Some of the elements of `ScenRedParms` are required (e.g. statistics for the input tree) while others are optional (e.g. the run time limit). SCENRED will stop if a required element is missing or out of range.

A few comments on the parameters `red_percentage` and `red_num_leaves` are in order. At least one of these values

Element	Description
num_leaves	the number of initial scenarios or leaves of the scenario tree (i.e., before the reduction)
num_nodes	number of nodes in the initial tree (the number of scenarios if not tree-structured)
num_random	Number of random variables assigned to a scenario or node, i.e., the dimension of the random data process
num_time_steps	Length of a path from the root node to a leaf of the scenario tree, i.e., the number of time steps involved

Table 39.1: Required ScenRedParms elements

Element	Description	Default
red_num_leaves	specifies the desired number of preserved scenarios or leaves	none
red_percentage	specifies the desired reduction in terms of the relative distance between the initial and reduced scenario trees (a real between 0.0 and 1.0)	none
num_stages	Set the number of branching levels of the scenario tree, i.e., the number of stages of the model -1. Hence num_stages=1 if no branching occurs, i.e., the values of the scenarios differ for all time steps.	1
where_random	An integer indicating where the randomness enters the model. The value is interpreted as a "digit map" computed using the formula *100*inObj + 10*inRHS + inMatrix*, where *inObj* is 1 if the objective contains random parameters and 0 otherwise, *inRHS* is 1 if the right-hand side contains random parameters and 0 otherwise, and *inMatrix* is 1 if the constraint matrix contains random coefficients and 0 otherwise.	10 (random right-hand side)
reduction_method	Select a reduction method: 0: automatic (best expected performance with respect to running time) 1: Fast Backward method 2: Mix of Fast Backward/Forward methods 3: Mix of Fast Backward/Backward methods	0
run_time_limit	Defines a limit on the running time in seconds	none
report_level	Control the content of the SCENRED log file: 0: Standard SCENRED log file 1: Additional information about the tree	0

Table 39.2: Optional ScenRedParms elements

must be set. The value of red_percentage will be ignored if the parameter red_num_leaves is non-zero. Otherwise, the tree will not be reduced if red_percentage=0, while the reduction of the tree will be maximal (i.e. only one scenario will be kept) if red_percentage=1. A numeric value of 0.5 means that the reduced tree maintains 50% of the information contained in the original tree. The reduction algorithms are skipped if red_num_leaves=num_leaves or if red_num_leaves=0 and red_percentage=0. These values can be assigned if the user wishes to run the scenario tree diagnostic.

The second data element in the input file is the set of nodes making up the scenario tree. Note that the cardinality of this set is part of ScenRedParms.

The third data element is the ancestor mapping between the nodes. This mapping determines the scenario tree. Note that the mapping can be either an ancestor mapping (i.e. child-parent) or a successor mapping (parent-child). By default, SCENRED expects an ancestor mapping. If the check for this fails, it looks for a successor mapping.

The fourth data element is the parameter of probabilities for the nodes in the original tree. It is only required that probabilities for the scenarios (i.e. the leaf nodes) be provided, but the parameter can contain probabilities for the non-leaf nodes as well.

The remaining elements in the input data file specify the parameter(s) that comprise the random values assigned to the initial scenarios, or to the nodes of the scenario tree. There can be more than one such parameter, included in any order. The only requirement is that the node set be the final index in each of these parameters.

Table 39.3 summarizes the content of the SCENRED input file. Please keep in mind that the order of the entries must not be altered!

No.	Symbol	Type	Dimension	Content
1	ScenRedParms	Parameter	1	scalar SCENRED input
2	(any name)	Set	1	nodes in the scenario tree
3	(any name)	Set	2	the ancestor set
4	(any name)	Parameter	1	node probabilities; at least for the leaves
$\geq$5	(any name)	Parameter	≥ 1	random values assigned to the nodes

Table 39.3: Content of the SCENRED Input File

To create the SCENRED data input file, the GAMS execute_unload statement is used. This statement is used to transfer GAMS data to a GDX file at execution time. As an example, to create a GDX file with the 4 required input parameters and one parameter demand containing the stochastic data, you might have the following statement:

```
execute_unload 'sr_input.gdx', ScenRedParms, node, ancestor, prob, demand
```

5 SCENRED Options and the Option File

When the SCENRED executable is run, it takes only one argument on the command line: the name of the SCENRED option file. The option file is a plain text file. Typically, it is used to specify at least the names of the SCENRED data input and output files. The option file must be created by the SCENRED user (typically via the GAMS put facility during the GAMS run). The syntax for the SCENRED option file is

```
optname value   or   optname = value
```

with one option on each line. Comment lines start with an asterix and are ignored.

Some of the SCENRED options may be specified in two places: as elements of the ScenRedParms parameter of the SCENRED input file, or as entries in the options file. These parameters have been summarized in Table 39.2. If an option is set in both these places, the value in the option file takes precedence of over the value from ScenRedParms. In addition, the parameters in Table 39.4 can only be specified in the option file.

Option	Description	Default
input_gdx	Name of the SCENRED data input file	xllink.gdx
output_gdx	Name of the SCENRED data output file	scenred.gdx
log_file	Name of the SCENRED log file	scenred.log

Table 39.4: Options - optfile only

6 The SCENRED Output File

The SCENRED output file contains the reduced scenario tree and the ScenRedReport parameter. Like the input file, the output file has a special binary format; it is a GDX (GAMS Data Exchange) file.

The first data element in the output file is the ScenRedReport parameter containing the scalar outputs and statistics from the SCENRED run. The elements of this parameter are summarized in Table 39.5. The second data element is the parameter containing the probabilities of the nodes in the reduced scenario tree. These node probabilities are required to construct the reduced tree. The third and final data element is the ancestor map for the reduced scenario tree. This map can be read from the GDX file, or the reduced tree can be built from the original one by using the reduced probablilities. The content of the data output file is summarized in Table 39.6.

Element	Description
ScenRedWarnings	number of SCENRED warnings
ScenRedErrors	number of SCENRED errors
run_time	running time of SCENRED in sec.
orig_nodes	number of nodes in the initial scenario tree
orig_leaves	number of leaves (scenarios) in the initial scenario tree
red_nodes	number of nodes in the reduced scenario tree
red_leaves	number of leaves(scenarios) in the reduced tree
red_percentage	relative distance of initial and reduced scenario tree
red_absolute	absolute distance between initial and reduced scenario tree
reduction_method	reduction method used: 0: the program stopped before it could select a method 1: Fast Backward method 2: Mix of Fast Backward/Forward methods 3: Mix of Fast Backward/Backward methods

Table 39.5: ScenRedReport elements

No.	Symbol	Type	Dimension	Content
1	ScenRedReport	Parameter	1	report of the SCENRED run
2	red_prob	Parameter	1	node probabilities for the reduced scenarios
3	red_ancestor	Set	2	the ancestor map for the reduced scenarios

Table 39.6: Content of the SCENRED Output File

To read the SCENRED data output file, the GAMS execute_load statement is used. This statement is used to transfer GDX data to GAMS at execution time. As an example, to read a GDX file named sr_output.gdx created by SCENRED, you might have the following statement:

```
execute_load 'sr_output.gdx', ScenRedReport, sprob=red_prob, sanc=red_ancestor
```

In the statement above, the equal sign "=" is used to indicate that the data in the GDX parameter red_prob should be read into the GAMS parameter sprob, and the data in the GDX set red_ancestor should be read into the GAMS set sanc.

7 Diagnostic Check of Scenario Trees

When SCENRED reads its input data, it performs a number of checks to verify that the data is correct. The diagnostic checks of the input parameters include:

- 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
- 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, check the consistency of the parameter `num_time_steps` with the ancestor matrix)

The following errors in the specification of the scenario tree cause SCENRED to skip the reduction algorithms:

- The input files cannot be opened.
- Not all required input parameters are given.
- The required input parameters are not consistent with the contents of the SCENRED input file.
- The required input parameters are out of range.
- Missing or negative scenario probabilities (probabilities of leaves).
- The ancestor set contains too many entries (more than `2*num_nodes`).
- SCENRED detects a cycle in the ancestor set.
- SCENRED detects incomplete scenarios in the ancestor set.
- Run time limit reached

8 SCENRED Errors and Error Numbers

When SCENRED encounters a serious error in the input files or in the scenario tree, it sends an error message to the screen and to the log file. These messages always start with

```
**** SCENRED run-time error ...
```

The number of SCENRED errors are contained in the parameter `ScenRedReport` of the SCENRED output file (if it could be created). The occurence of an error can also be detected from the last line that SCENRED sends to the screen:

```
**** SCENRED ErrCode=...
```

The numerical values of `ErrCode` and their meaning are given below.

ErrCode	Meaning
1	(for internal use)
2	fatal error while reading from SCENRED input file
3	fatal error while writing to SCENRED output file
4	fatal error while reading from SCENRED option file
5	log file cannot be opened
6	a memory allocation error occured
7	there are missing input parameters
8	could not access the GAMS names for the nodes
9	(for internal use)
10	ancestor set not given or contains too many entries
11	node probabilities cannot be not read or are wrong
12	random values for the nodes cannot be read
13	input parameters are out of range
14	ancestor set contains a cycle
15	incomplete scenarios or forests detected
16	fatal error in reduction algorithm (not enough memory)
17	running time limit reached

9 SCENRED Warnings

SCENRED warnings are caused by misspecification of the initial scenarios that can be possibly fixed. When SCENRED encounters such an error in the input files or in the scenario tree, it sends a message to the screen and to the log file. These messages always start with

```
**** SCENRED Warning ...
```

The following list gives an overview of the cases that produce warnings, and the action taken by SCENRED in these cases.

- The user assigned an option value that is out of range.
  Action: Assign the default value.
- Both parameters `red_num_leaves` and `red_percentage` are assigned nontrivial values.
  Action: The value of `red_percentage` will be ignored.
- The scenario probabilities (probabilities of leaves) do not sum up to 1.
  Action: The scenario probabilities are rescaled. Assign new probabilities to the remaining (inner) nodes that are consistent with the scenario probabilities.
- Missing probabilities of inner nodes.
  Action: Assign node probabilities that are consistent with the scenario probabilities.
- The ancestor set contains more than one ancestor for a node.
  Action: SCENRED assumes to be given a successor set instead of an ancestor set (i.e., the transpose of an ancestor matrix. This means that the graph corresponding to the ancestor set has the wrong orientation). SCENRED starts the tree diagnostic for the successor set. The reduced tree will be defined in terms of a successor set as well (if the successor set passes the tree diagnostic and if SCENRED locates no fatal error during the run).
- The fast backward method delivered a result, but the result cannot be improved by the forward or backward method (running time limit reached).
  Action: Use the result of the fast backward method.