SCENRED

Contents

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`)

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

2. 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.

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 1.1: 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 1.2: Optional `ScenRedParms` elements

A few comments on the parameters `red_percentage` and `red_num_leaves` are in order. At least one of these values 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 1.3 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 1.3: 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 1.2. 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 1.4 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 1.4: 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 1.5. 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 1.6.

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 1.5: `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 1.6: 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.

SCENRED–2

Contents

1 Introduction

Scenred2 is a fundamental update of the well-known scenario reduction software Scenred. A lot of new features come along with the latest release version. Beside updates and extentions concerning the control of the scenario reduction action an all new device for scenario tree construction has been implemented in Scenred2. Moreover, a lot of visualization functions to plot scenario trees and scenario processes linked to the free Gnuplot plotting software are available with Scenred2 now.

Table: Summary of basic new functions in Scenred2

Description	Section
Additional options for controlling the scenario reduction	3
New device of scenario tree construction	4
Visualization of scenario trees and processes	5
Command line interface and data export	6

2 Using Gams/Scenred2

Successful applying Scenred or Scenred2 requires a special formulation of the stochastic programming model within the Gams program. Probabilistic information must be given by a set of nodes implying a certain ancestor structure including a well-defined root node. Note that the usage of Gams/Scenred2 is basically the same as the usage of Gams/Scenred. Hence, it is recommended for new users to look at the Scenred documentation first. All details about how to organize your Gams program, how to run Scenred from the Gams program by using the gdx interface, and, of course, examples can be found in that documentation.

The Gams/Scenred2 link provides the same gdx interface. But, due to new features some small changes in controlling the options are needed. Scenred2 supports now two types of option files. The first one is the SR-Command-File which must be passed to Scenred2 together with the Scenred2 call. The second one, the SR-Option-File includes more specific options to control the selected scenario reduction or scenario construction methods and can be declared in the SR-Command-File.

SR-Command-File

The command file includes the basic specifications. These are input/output gdx file names, the log file name, all other file names which are needed for diverse visualization and output options. It also includes the name of the SR-Option-File.

Table: Supported options of SR-Command-File

Option	Description	Required
`log_file`	specify a log file name	yes
`input_gdx`	specify the gdx input file for Scenred	yes
`output_gdx`	specify the gdx output file of Scenred	yes
`sr_option`	specify a SR-Option-File	no
`visual_init`	specify a name for visualization of input tree	no
`visual_red`	specify a name for visualization of reduced/constructed tree	no
`plot_scen`	specify a name for visualization of scenario processes	no
`out_scen`	specify a file for scenario data output in fan format	no
`out_tree`	specify a file for scenario data output in tree format	no

Example:

Scenred2 must be called together with a command file, which contains at least all required options. The data exchange via the gdx interface and the Scenred2 call from the Gams program is of the form (be careful with the meanings and right order of gdx symbols):

```
execute_unload 'srin.gdx', ScenRedParms, n, ancestor, prob, random;
execute 'scenred2 scenred.cmd';
execute_load 'srout.gdx', ScenRedReport, ancestor=red_ancestor, prob=red_prob;
```

For example, the command file could be the following (note the compatible gdx file names):

```
* scenred command file 'scenred.cmd'

log_file    sr.log
input_gdx   srin.gdx
output_gdx  srout.gdx
sr_option   scenred.opt
visual_red  tree
out_scen    raw.dat
```

ScenRedParms

With the symbol list of the parameter ScenRedParms and the SR-Option-File all necessary information regarding the Scenred2 run can be assigned. The Gams parameter ScenRedParms can easily included to the Gams program by the statement:

```
$libinclude scenred2
```

Of course, the include must be stated before calling Scenred2. After that statement all supported parameters can be assigned, but at least all required parameters regarding the input scenarios. By the symbols of the parameter ScenRedParms you make also the decision of what features you exactly want to use with Scenred2. Moreover, some other usefull parameters for the Scenred2 run are included in the symbol list of the parameter ScenRedParms.

Table: Supported Scenred2 parameters in ScenRedParms

Symbol	Description	Required
`num_time_steps`	path length from root to leaf	yes
`num_leaves`	leaves/scenarios in the initial tree	yes
`num_nodes`	nodes in the initial tree	yes
`num_random`	random variables assigned to a scenario or node	yes
`red_num_leaves`	desired number of preserved scenarios or leaves	no
`red_percentage`	desired relative distance (accuracy)	no
`reduction_method`	desired reduction method	no
`construction_method`	desired tree construction method	no
`num_stages`	number stages	no
`run_time_limit`	time limit in seconds	no
`report_level`	report level: more messages by higher values	no
`scen_red`	scenario reduction command	no
`tree_con`	tree construction command	no
`visual_init`	visualization initial tree	no
`visual_red`	visualization reduced (constructed) tree	no
`plot_scen`	visualization scenario processes	no
`out_scen`	output of scenario raw data	no
`out_tree`	output of scenario tree data	no

To enable some options assign a value to the parameter. A parameter value of zero (default) disables an option. Note that when running Scenred2 either scenario reduction or scenario tree construction can be performed. Hence, only `scen_red` or `tree_con` should be used at once.

<u>Example:</u>

The following statements describe a possible example setup for proceeding the scenario tree construction with visualization of the scenario tree and output of the scenarios to a raw data file afterwards. Note that for the visualization and the scenario output the name of output files must be specified in the SR-Command-File. Otherwise a warning will inform you about not selected file names.

```
* general parameters

ScenRedParms('num_leaves') = 100;
ScenRedParms('num_nodes') = 200;
ScenRedParms('num_time_steps') = 5;
ScenRedParms('num_random') = 2;
ScenRedParms('report_level') = 2;
ScenRedParms('run_time_limit') = 30;

* execution commands

ScenRedParms('tree_con') = 1;
ScenRedParms('visual_red') = 1;
ScenRedParms('out_scen') = 1;
```

Scenred2 can also be used for plotting tasks only. Disable both the `scen_red` and `tree_con` option and use one ore more visualization options only (see also Section 5 for more details regarding visualizations).

SR-Option-File

The SR-Option-File is the more specific option file and will be passed to Scenred2 by the `sr_option` statement specified in the SR-Command-File. It serves as control unit for available methods provided by Scenred2. The supported options depend on what kind of method is called with Scenred2. A detailed list of all options together with examples are given below for both the scenario reduction and the scenario construction devices (see Sections 3 and 4, respectively). Note that certain parameters can be assigned by using both ScenRedParms and the SR-Option-File. In case of having parameters defined twice a warning by Scenred2 will generated to inform you.

3 Scenario Reduction

The scenario reduction device consists of approved methods for reducing the model size by reducing the number of scenarios in an optimal way. Here it doesn't make any difference whether the input data is structured as scenario tree or not. But note, the classical scenario reduction approach is actually developed for two-stage models. Extensions for the multistage case are planed in the near future. To learn more about the mathematical theory see recent publications, for example [5, 4, 2].

With Scenred2 the most popular and accurate reduction algorithms of forward and backward type are maintained further on. New options make it possible to proceed with the scenario reduction more individual. The most important new parameter is given by the option `metric_type` which allows to control the reduction process by different type of probability distances. Altogether three distances can be selected (see Table below). All probability distances are associated with a special order specification which can be set by the new option `order`. Both options replace the old option `where_random` which is not used any longer.

Table: SR Options – Scenario Reduction

Option	Description
`red_num_leaves`	desired number of scenarios (integer)
`red_percentage`	relative accuracy (number from 0.0 to 1.0)
`reduction_method`	1 - Forward, 2 - Backward, 0 - Default
`metric_type`	1 - Transport (default), 2 - Fortet-Mourier, 3 - Wasserstein
`p_norm`	choice of norm (example: 0 - max, 1 - sum, 2 - Euclidian)
`scaling`	0 - scaling off, 1 - scaling on (default)
`order`	metric order (integer, default is 1)

Example:

For example, a valid SR-Option-File is the following:

```
* scenred option file

reduction_method  1
red_percentage    0.3
metric_type       2
order             2
p_norm            1
scaling           0
```

Lines starting with the star symbol (route symbol can be used too) provide comment lines. The star symbol can also be used to out comment and disable certain options.

4 Scenario Tree Construction

Scenario tree construction is the outstanding all new device of Scenred2. It allows to construct scenario trees as accurate input for multistage stochastic programs (cf. [3]). The input are individual scenarios in form of a fan which must be allocated before calling Scenred2. A lot of options are offered to control the tree construction process. Note that in some cases due to sensibility of certain parameters some tuning is indispensable for producing good results.

Table: SR Options (basic) – Scenario Tree Construction

Option	Description
`construction_method`	1 - forward, 2 - backward
`reduction_method`	1 - forward, 2 - backward, used within the iteration
`first_branch`	time period of first branch (integer)
`red_percentage`	relative accuracy (level from 0.0 to 1.0)
`eps_growth`	1 - linear, 2 - exponential
`eps_evolution`	tree structure parameter (from 0.0 to 1.0)
`scaling`	0 - scaling off, 1 - scaling on (default)
`order`	order of metric

The Table above displays the main options to control the tree construction process. They are very simular to the reduction options. The role of the option `red_percentage` is here to prescribe a total epsilon accuracy (level) for the approximation scheme. But the approximation scheme is based on stagewise approximations which requires a splitting of the total level to the stages. Two strategies are offered by Scenred2 a linear and an exponential mapping of the total level to the intermediate levels. Use option `eps_growth` to select one of them. Both strategies allow a second tuning parameter `eps_evolution` which effects the slope of the epsilon splitting.

Even though this kind of control may generate good results for many applications, sometimes a more individual control can be needed. For example, some applications require a localization of branching stages. Moreover, to setup approximation bounds directly to stages can be very useful. To this end the standard options are extended by a new section environment.

Additional options – The section environment

An alternative control for the accurate constructions is provided by using the section environment. The section environment aims to establish a better monitoring of the construction process. There are overall three section types supported by Scenred2 with the same syntax.

Branching control:

This section environment allows to specify branching points, i.e., an explicit selection of stages serving for branching. For example, use

```
section branching
  2
  4
  6
end
```

to allow branching only at time period 2, 4, and 6. Note that each stage statement must be placed in one line. But stages can be merged. A shorter formulation of the same contents can be written in closed form

```
section branching
```

```
  2*6  2
end
```

This statement reads branching within time periods from period 2 to period 6 with increment 2 steps. Both assignments can be combined and should be used together with the `red_percentage` option.

Epsilon control:

In the simular way by the epsilon section it is possible to assign epsilon tolerances for the stage approximations explicitly. This environment overcomes difficulties at times coming across with the automatic epsilon control. Note that this environment disables the option `red_percentage`. For example, use

```
section epsilon
  2    0.04
  3*4  0.03
  5    0.02
  6    0.01
end
```

to control the approximation scheme by assigning different epsilon values per stage. Note that the value 0.03 is assigned to time period 3 and 4 in the example.

Node control:

The node control is the most specific control you have over the tree construction. With this environment the number of nodes of the tree which will generated can be assigned for each time stage explicitly. For example, use

```
section num_nodes
  1    1
  2*3  5
  4*5  10
  6    15
end
```

The syntax is the same as before. Note that only one section environment can be use at once. In particular, only the first section environment detected in the option file is used. The section environment can be out commented like a standard option too.

Experimental option:

There is one other useful option to speed up computations when building different scenario trees from exactly the same input data. In this case the scenario distances needed to compute the trees could be saved to a external file at the first run and reloaded at later runs. Hence, the distances must be calculated only once. For example, use the option

```
write_distance  dist.sr2
```

to save the computed distances to the file 'dist.sr2'. To reload them at next run use the option

```
read_distance  dist.sr2
```

The option is classified as experimental since no validation of the input file takes place. Before using this option, please ensure that the distances loaded with the `read_distance` option are the right ones.

Example:

Finally, look at the following example to see a valid SR-Option-File which can be passed to the scenario tree construction:

```
* tree construction option file

construction_method  2
reduction_method     1
order                1
scaling              0

section epsilon
  2*4    0.1
  5      0.2
  6      0.1
end
```

Example problem 'srpchase.gms'

A small example problem has been included to the GAMS Model Library. The implementation can be found in the Gams program 'srpchase.gms'. It might help you to practice in building scenario trees using Gams/Scenred2. The problem is to solve a simple stochastic purchase problem involving three stages. Sample scenarios which are generated from a fixed distribution using a random value generator serve as input for the tree construction.

5 Visualization

Visualization is another all new feature of Scenred2. In this section an easy way for making plots of scenario processes and tree structures is described. To this end you need the free Gnuplot software or any other plotting software which allows plotting directly from simple data files.

The concept of plotting tasks is the following. For each plot two files are generated, a Gnuplot access file (name.plt) and a raw data file (name.dat). The access file contains basic Gnuplot options and it can be adjusted for individual liking afterwards. The default output is the display. The supported plotting commands are

```
visual_init, visual_red, plot_scen
```

for plotting the initial tree structure, the reduced/constructed tree structure, and the scenario process(es), respectively.

Example:

For example, to visualize the constructed tree use the option

```
visual_red tree
```

within the SR-Command-File to specify the name for the output and activate the ScenRedParms parameter

```
ScenRedParms('visual_red') = 1;
```

in the Gams program. The result are the output files 'tree.plt' and 'tree.dat'. To compute the picture now you simply open the file 'tree.plt' with Gnuplot from the directory, where both output files are located (that should be the working directory). Alternatively, from the command line prompt call

```
>gnuplot tree.plt
```

Gnuplot will automatically generate the picture. Feel free to change any option in the Gnuplot access file for individual requirements. See also the Gnuplot manual for more details. In particular, to compute a well-scaled encapsulated postscript picture (eps), you simply have to uncomment a few lines in the Gnuplot option file above and to open it with Gnuplot once again.

With the command `plot_scen` the scenario process(es) can be visualized. Note that Scenred2 generates Gnuplot access and data files according to the number of random values.

6 Command Line Interface

The command line interface allows to run Scenred2 stand alone without using Gams. In this case the input for scenario reduction and scenario tree construction is handled by special input data files. The command file will be extended by the parameters having with the ScenRedParms otherwise.

To execute Scenred2 from the command line prompt together with a specified command file (which is required again), for example, call

```
>scenred2 command.file -nogams
```

To avoid diverse error messages do not forget the '-nogams' option to switch off the Gams interface. The command file can include some of the following options.

```
report_level  <integer>
runtime_limit <integer>
read_scen     <input file>
scen_red      <option file>
tree_con      <option file>
visual_init   <name>
visual_red    <name>
plot_scen     <name>
out_scen      <file name>
out_tree      <file name>
```

The denotation is not accidental the same as in case of using the Gams interface. The meaning of a certain option is maintained for the command line interface. To compute any scenario reduction or scenario tree construction the same SR-Option-Files are supported. It remains to clarify the data input format which comes across with the new `read_scen` command.

Data input format

To feed Scenred2 with data the scenario parameters must be passed by the `read_scen` command. Two types of input file formats are accepted.

a) The tree format:

This file is a formated data file including all information of the input scenarios tree. It must have a header with dimension information and the scenario data separated for each node. The header includes the type declaration, the number of nodes, and the number of random values.

The data part starts with the key word `DATA` (do not forget). The tree data has to be ordered node by node. For every node the following information is expected separated by white spaces: The unique predecessor node (root node points to itself) followed by the node probability and followed by the assigned number of random data values.

All information to one node should be written to one line (only for clearness reasons). Comment lines are allowed.

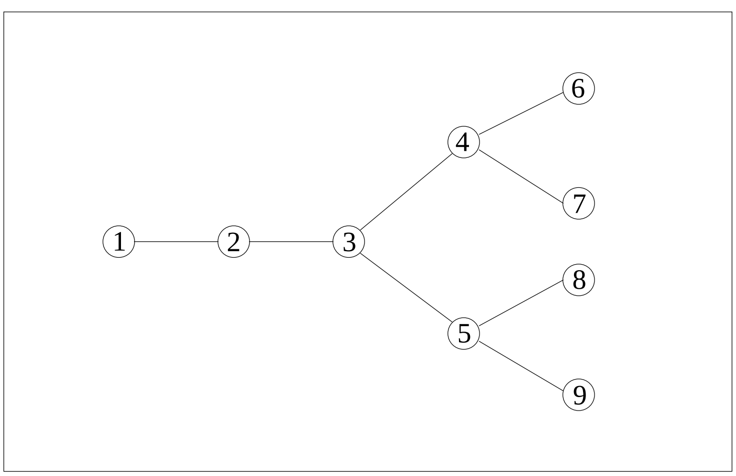
Match the following conventions:

– Nodes are identified by a sequence of integer numbers.
– The root node is expected to be the node '1'.
– The predecessor of root is '1' too, i.e., the root points to itself.
– All nodes numbers require a canonical order by stages and scenarios (see example).

Example:

```
# input tree example for scenred

TYPE TREE

NODES   9
RANDOM  4

DATA
* PRED PROB  RAND-1 RAND-2 RAND-3 RAND-4
   1    1.0    42.5    9.1    7.5  120.0
   1    1.0    39.8   11.2    8.4   90.0
   2    1.0    37.6   14.0    6.3  110.0
   3    0.5    38.9   12.4    8.1  130.0
   3    0.5    35.7   13.8    7.5  120.0
   4   0.25    40.3   14.9    7.2  120.0
   4   0.25    38.4   15.2    8.9  100.0
   5    0.3    37.6   14.9    9.3   80.0
   5    0.2    36.3   12.8   10.3   90.0
END
```

Figure: The scenario structure of the example tree



b) The fan format:

A scenario fan serves as input for the scenario tree construction but it can be used also for the scenario reduction. The scenario fan represents a special form of a scenario tree, where we consider individual scenarios merged to a collective root node (the root node can also be viewed here as some kind of artificial node).

Accordingly, the fan input file is a formated data file including all information of the scenarios in individual form now. It must have a simular header with dimension information and the scenario data separated now for each scenario. The header gets the type declaration `FAN` instead of `TREE` and includes the number of scenarios, the number of time periods, and the number of random values. The data part is opened again with the `DATA` key word.

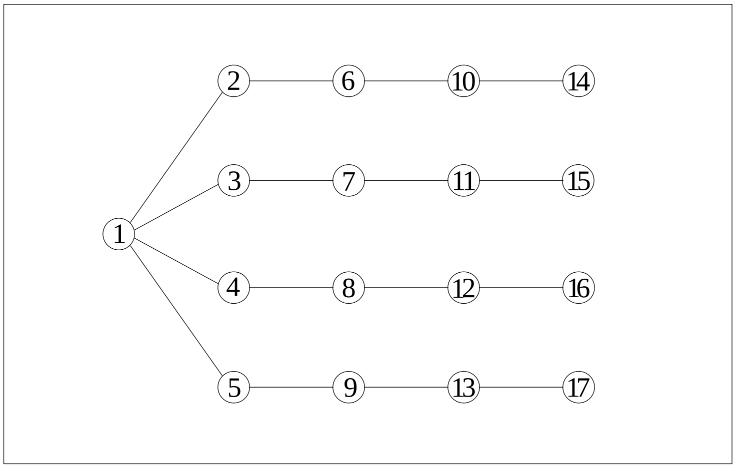
Every scenario is specified by a dataset including the scenario probability first followed by the different random values in ascending order w.r.t. time periods. All entries must be separated by a white space. Comment lines can be placed by the star and route symbols again. Note that in case of having an undetermined root node the mean of random values will taken for the first time period to appoint a unique root node. The example tree represented as input in scenario fan format is displayed in the next example.

Example:

```
# input fan example for scenred

TYPE  FAN

TIME    5
SCEN    4
RANDOM  4

DATA
0.2500
42.5      9.1      7.5     120.0
39.8     11.2      8.4      90.0
37.6     14.0      6.3     110.0
38.9     12.4      8.1     130.0
40.3     14.9      7.2     120.0

0.2500
42.5      9.1      7.5     120.0
39.8     11.2      8.4      90.0
37.6     14.0      6.3     110.0
38.9     12.4      8.1     130.0
38.4     15.2      8.9     100.0

0.3000
42.5      9.1      7.5     120.0
39.8     11.2      8.4      90.0
37.6     14.0      6.3     110.0
35.7     13.8      7.5     120.0
37.6     14.9      9.3      80.0

0.2000
42.5      9.1      7.5     120.0
39.8     11.2      8.4      90.0
37.6     14.0      6.3     110.0
35.7     13.8      7.5     120.0
36.3     12.8     10.3      90.0
END
```

Note that even though all scenarios coincide at the first three time periods, in this example, the scenarios will represented by one node each for every time period by the fan input format. The exception is the first time period, where a unique root node is expected in general and, therefore, only one node is assigned. The following picture shows the structure of the scenario fan which is generated by the example input.

Figure: The structure of the example input in fan format



Data Export

Scenred2 allows to export scenario data after computing the scenario reduction or scenario tree construction to external data files. Data export is available for both the Gams and the command line interface. To export data from Scenred2 two output options `out_tree` and `out_scen` can be use. These options generate data files according to the tree and fan format, respectively. The name of the data files will be specified in the SR-Command-File. When using the Gams interface the options must be connected by activating the corresponding ScenRedParms parameter, additionally.

7 A Simplified Interface to Scenred2: `$libinclude runscenred2`

While the previously described interface between GAMS and Scenred2 provides a maximum of flexibility, it also is rather complex and error-prone. The GAMS utility `runscenred2` tries to hide most of the mechanics of the GAMS/Scenred2 interface. The call to `runscenred2` looks as follows:

```
$libinclude runscenred2 myprefix tree_con n tree p rtree rp rv1 rv2
```

Table: runscenred2 Arguments:

	Argument	Description
1	`myprefix`	base name for files used with Scenred2
2	`tree_con` or `scen_red`	select Scenred2 action: tree construction or scenario reduction
3	`n`	the set of nodes in the tree
4	`tree`	the set of ancestor relations describing the tree
5	`p`	the parameter containing the node probablities
6	`rtree`	the set of ancestor relations of the reduced tree (output)
7	`rp`	the parameter containing the node probablities for the reduced tree (output)
8-	`rv1`, `rv2`, ...	parameters containing random values of the nodes

The table above describes the arguments of the `runscenred2` call. Arguments 3, 4, 5, 8 and following correspond to the symbols that need to be exported to the Scenred2 data input file (done with the `execute_unload` call in the complex interface). The output arguments 6 and 7 correspond to the symbols imported from te Scenred2 data output file (done with the `execute_load` call in the complex interface). The parameters `ScenRedParms` and `ScenRedReport` are invisibly communicated with Scenred2.

The second argument instructs Scenred2 either to construct a tree (`tree_con`) or to reduce a tree (`scen_red`).

Instead of providing an explicit name for all the different files in the Scenred2 command file, the first argument determines the name of all files using a simple naming scheme. The following name scheme is observed:

Filename	Command option	Description
`sr2`*myprefix*`.log`	`log_file`	log file name
`sr2`*myprefix*`_in.gdx`	`input_gdx`	gdx input file name
`sr2`*myprefix*`_out.gdx`	`output_gdx`	gdx output file name
`sr2`*myprefix*`.opt`	`sr_option`	option file name
`sr2`*myprefix*`_vi.plt`	`visual_init`	file name for visualization of input tree
`sr2`*myprefix*`_vr.plt`	`visual_red`	file name for visualization of reduced/constructed tree
`sr2`*myprefix*`_plot.plt`	`plot_scen`	file name for visualization of scenario process
`sr2`*myprefix*`_raw.dat`	`out_scen`	file name for scenario data output in fan format
`sr2`*myprefix*`_tree.dat`	`out_tree`	file name for scenario data output in tree format

The first three files (`log_file`, `input_gdx` and `output_gdx`) are always used. The only optional input file `sr_option` is read by Scenred2 if `ScenRedParms('sroption')=1`. When you create this file, make sure to use the proper file name. The output files are created by Scenred2 if the corresponding option is set to 1 in `ScenRedParms`, e.g. `ScenRedParms('out_tree')=1`.

In addition to a simpler communication of data between GAMS and Scenred2, the newer versions of GAMS/Scenred2 (starting with GAMS distribution 23.1) release the user of setting required fields in the `ScenRedParms` parameter: `num_time_steps`, `num_leaves`, `num_nodes`, and `num_random`. GAMS/Scenred2 calculates these numbers from its input data. In case the user still sets these fields, Scenred2 will ensure that the internbally calculated numbers and the user provided numbers match.

References

1. Heitsch, H.: Stabilität und Approximation stochastischer Optimierungsprobleme, dissertation, Logos Verlag Berlin, 2007.

2. Heitsch, H.; Römisch, W.: Scenario tree reduction for multistage stochastic programs, *Computational Management Science* 6 (2009), 117–133.

3. Heitsch, H.; Römisch, W.: Scenario tree modeling for multistage stochastic programs, *Mathematical Programming* 118 (2009), 371–406.

4. Heitsch, H.; Römisch, W.; Strugarek, C.: Stability of multistage stochastic programs, *SIAM Journal on Optimization* 17 (2006), 511–525.

5. Heitsch, H.; Römisch, W.: A note on scenario reduction for two-stage stochastic programs, *Operations Research Letters* 35 (2007), 731–738 .

6. Heitsch, H.; Römisch, W.: Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications* 24 (2003), 187–206.