User's Guide to `ddsip` – A C Package for the Dual Decomposition of Two-Stage Stochastic Programs with Mixed-Integer Recourse

A. Märkert, R. Gollmer
Department of Mathematics
University Duisburg-Essen, Campus Essen
Thea-Leymann-Straße 9, D-45127 Essen, Germany
ralf.gollmer@uni-due.de

19. August 2018

1 Introduction

`ddsip` is a C-implementation of a number of scenario decomposition algorithms for two-stage stochastic linear programs with mixed-integer recourse and/or mixed-integer first stage. It expects a MIP, stochastic LPs are not supported.

The program is based on a previous Fortran 90-implementation of C.C. Carøe. The main idea of the decomposition algorithms is the Lagrangian relaxation of the nonanticipativity constraints and a branch-and-bound algorithm on the first-stage variables to reestablish nonanticipativity. This branch-and-bound and the dual nonsmooth optimization are complemented by the addition of Benders feasibility cuts when heuristic solutions derived in the course of the solution process from the single scenario problems are infeasible for one of the scenarios.

The original scenarios decomposition algorithm has been developed in Carøe and Schultz (1999). Extensions including the treatment of mean-risk models have been made in Märkert (2004).

For the dual optimization we use `ConicBundle` – a C^{++}-implementation kindly provided by C. Helmberg, see Helmberg (2012).
We use the CPLEX callable library to solve the mixed-integer subproblems in the branch-and-bound tree, see CPLEX (2016). The current version of `ddsip` is ready for CPLEX 12.7.1 and requires at least version 11.2.

The implementation features risk minimization, too. This version supports mean-risk models involving the risk measure *expected excess of a target* (see Ogryczak and Ruszczynski (1999), here: *expected shortfall below target*), *excess probabilities* (as investigated in Schultz and Tiedemann (2003)), *absolute semideviation*, *worst-case-costs*, *tail value-at-risk*, *value-at-risk*, and *standard deviation.*

The code is of research quality, i.e. no production quality should be expected with respect to stability and efficiency. We kindly ask the user to support us fixing bugs by reporting them to us as they occur.
We did not include support for the SMPS format, but a rather basic input format for the **two-stage scenario formulation**. Any contributions in this respect are welcome.
This manual describes the format of the input files and the data contained in the output files of `ddsip`. We try to provide all necessary information on the input parameters.

2 Stochastic programs with mixed-integer recourse

This implementation is appropriate to solve *recourse models.* Such problems were first investigated by Dantzig (1955) and Beale (1955). The conceptual idea behind recourse models is the following; assume the decisions are two-stage in the sense that some of them, say x, have to be taken immediately whilst others, say y, may be delayed to a time when uncertainty has revealed. We can write a random linear program of this type as

$$\inf_{x\in X,\, y(\omega)\in\mathbb{R}^m_+}\{c(\omega)x + q(\omega)y(\omega) \ :\ T(\omega)x + W(\omega)y(\omega) = h(\omega)\}. \tag{1}$$

The random parameter $\xi := (c, q, T, W, h) : \Omega \to \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{s\times n} \times \mathbb{R}^{s\times m} \times \mathbb{R}^s$ is defined on the probability space $(\Omega, \mathbb{P}, \mathcal{A})$. The set $X \subset \{x \in \mathbb{R}^n_+ : Ax = b\}$ contains all deterministic constraints on x. Inequality constraints can be handled in (1) by the introduction of appropriate slack variables. The model (1) is also referred to as *two-stage model.*
We note that the problem (1) is not yet well-defined. As the constraints include random parameters, the meaning of feasibility and thus of optimality is not clear. We complete the recourse model by adding an objective function criterion. Before we do so, we rewrite problem (1) as

$$\inf_{x\in X}\{c(\omega)x + \phi(x,\omega)\} \tag{2}$$

where

$$\phi(x,\omega) = \inf_{y\in\mathbb{R}^m_+}\{q(\omega)y \ :\ T(\omega)x + W(\omega)y = h(\omega)\}. \tag{3}$$

We note that, provided $\phi : \mathbb{R}^n \times \Omega$ is measurable, we can regard $\mathcal{Z} := \{c(\omega)x + \tilde{\phi}(x,\omega) : x \in X\}$ as a family of random variables. Now, each function $\mathcal{R} : \mathcal{Z} \to \mathbb{R}$, e.g. the expected value, some measure of risk, or a weighted sum of both, can serve as objective criterion

$$\inf_{x\in X} \mathcal{R}[c(\omega)x + \tilde{\phi}(x,\omega)]. \tag{4}$$

Our focus is on integer models, i.e. in addition to the constraints employed in problem (1) we may have integrality requirements on the variables x and y. Note that models with a linear second stage should be dealt with a version of the *L-shaped algorithm*, see Birge and Louveaux (1997).

We briefly discuss the implemented algorithm for the expected value case, i.e.

$$\mathcal{R}[c(\omega)x + \tilde{\phi}(x,\omega)] = \mathrm{E}[c(\omega)x + \tilde{\phi}(x,\omega)] := \int_{\Omega} c(\omega)x + \tilde{\phi}(x,\omega)\mathbb{P}d\omega.$$

Further algorithms suitable for the treatment of mean-risk models are described in Märkert (2004).
Assume we are given a finite number of scenarios ξ_j, $j = 1,\ldots,S$, with corresponding probabilities π_j. Then, problem (4) turns into

$$\min_{x\in X,\, y_j\in\mathbb{Z}_+^m\times\mathbb{R}_+^{m'}} \{c_j x + \sum_{j=1}^{S} \pi_j q_j y_j \ : \ T_j x + W_j y_j = h_j, \quad \forall j\}, \tag{5}$$

cf. Birge and Louveaux (1997), Kall and Wallace (1994), and Prekopa (1995). The so-called expected recourse function Q_{E} reads

$$Q_{\mathrm{E}}(x) = c_j x + \min_{y_j\in\mathbb{Z}_+^m\times\mathbb{R}_+^{m'}} \{\sum_{j=1}^{S} \pi_j q_j y_j \ : \ T_j x + W_j y_j = h_j, \quad \forall j\},$$

for all $x \in \mathbb{R}^n$. The program (5) is a large-scale deterministic mixed-integer linear program (MILP) with a block-angular structure. A recent and comprehensive overview of existing algorithms for problem (5) is provided in Louveaux and Schultz (2003). To mention some of the algorithmic approaches we refer to van der Vlerk (1995) for simple recourse models ($W = (I, -I)$), to Laporte and Louveaux (1993) for two-stage models with a binary first stage, and to Ahmed et al. (2000) for models with an integer second stage and a fixed technology matrix T. An algorithm for problem (5) in its general form has been proposed in Carøe and Schultz (1999). It works on the expense of a branching on continuous first-stage variables. The latter algorithm is implemented in `ddsip` and described in what follows.
By introducing copies of the first-stage variables, an equivalent formulation of (5) is given by

$$\min_{x_j,y_j}\{\sum_{j=1}^{S} \pi_j(c_j x_j + q_j y_j) \ : \ x_1 = \ldots = x_S,\, (x_j, y_j) \in M_j,\, \forall j\} \tag{6}$$

where $M_j = \{(x_j, y_j) : T_j x_j + W_j y_j = h_j, x_j \in X, y_j \in Y\}$, $j = 1,\ldots,S$.
Considering the constraint matrix of (6) (cf. Figure 1), we can identify S single-scenario subproblems solely coupled by the equality (*nonanticipativity*) constraints on the copies of the first-stage variables and written as $\sum_{j=1}^{S} H_j x_j = 0$, where $H = (H_1,\ldots,H_j)$. The problem decomposes when we relax the nonanticipativity constraints.
Upper bounds on the optimal value can be obtained by heuristics based on the solutions for the subproblems. We get a lower bound by solving the Lagrangian dual, which is a nonlinear concave maximization

$$z_{\mathrm{LD}} := \max_{\lambda\in\mathbb{R}^l} \min_{x_j,y_j}\{\sum_{j=1}^{S} \pi_j(c_j x_j + q_j y_j) + \lambda \sum_{j=1}^{S} H_j x_j \ : \ (x_j, y_j) \in M_j,\, \forall j\}. \tag{7}$$

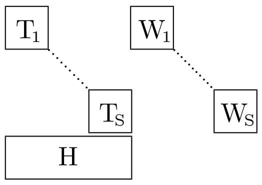

Abbildung 1: Constraints of (6)

In general, the involved integrality restrictions lead to an optimality gap. If we are not satisfied with the bounds given by the above method, we can elaborate a branch-and-bound algorithm that successively reestablishes the equality of the components of the first-stage vector. Let $\mathcal{P}$ denote a list of problems.

`Algorithm SD`: Scenario decomposition (Carøe and Schultz (1999))

STEP 1 Initialization: Set $z^* = \infty$ and let $\mathcal{P}$ consist of problem (5).

STEP 2 Termination: If $\mathcal{P} = \emptyset$ then x^* with $z^* = Q_{\mathrm{E}}(x^*)$ is optimal.

STEP 3 Node selection: Select and delete a problem P from $\mathcal{P}$ and solve its Lagrangian dual. If the associated optimal value $z_{\mathrm{LD}}(P)$ equals infinity (infeasibility of a subproblem) go to *STEP 2*.

STEP 4 Bounding: If $z_{\mathrm{LD}}(P)$ is greater than z^* go to *STEP 2*. Otherwise proceed as follows; if the first-stage solutions x_j, $j = 1, \dots, S$, of the subproblems are

- identical, then set $z^* := \min\{z^*, Q_{\mathrm{E}}(x_j)\}$, delete all $P' \in \mathcal{P}$ with $z_{\mathrm{LD}}(P') \geq z^*$ and go to *STEP 2*.
- not identical, then compute a suggestion $\hat{x} := Heu(x_1, \dots, x_S)$ using some heuristic. Set $z^* := \min\{z^*, Q_{\mathrm{E}}(\hat{x})\}$ and delete all $P' \in \mathcal{P}$ with $z_{\mathrm{LD}}(P') \geq z^*$.

STEP 5 Branching: Select a component $x_{(k)}$ of x and add two new problems to $\mathcal{P}$ that differ from P by the additional constraint $x_{(k)} \leq \lfloor x_{(k)} \rfloor$ and $x_{(k)} \geq \lfloor x_{(k)} \rfloor + 1$, respectively, if $x_{(k)}$ is integer, or $x_{(k)} \leq x_{(k)} - \varepsilon$ and $x_{(k)} \geq x_{(k)} + \varepsilon$, respectively, if $x_{(k)}$ is continuous. $\varepsilon > 0$ has to be chosen such that the two new problems have disjoint subdomains. Go to *STEP 3*.

The algorithm is finite if X is bounded and if some stopping criterion is employed that prevents the algorithm from endless branching on the continuous components of x, see Carøe and Schultz (1999).
The function Q_{E} is evaluated at x by fixing the first stage to x, solving the scenario many subproblems, and calculating the expected value of the corresponding optimal values. Thus, infeasible suggestion are identified immediately.
We remark that problems related to random recourse have to be cared about by the users, cf. Walkup and Wets (1967).

3 Input files

`ddsip` requires 3 input files, a number of other files are optional.

- Model file: a file readable by CPLEX (e.g. lp- or mps-format, possibly in gzipped form) that specifies the single-scenario model
- Specification file: a file containing the specifications of the stochastic program, some CPLEX- and b&b-parameters
- Priority order file for subproblems (optional): a CPLEX order file corresponding to the model file
- RHS scenario file: a file containing the stochastic right-hand sides and the probabilities of the scenarios
- Cost scenario file (optional): a file containing stochastic cost coefficients
- Matrix scenario file (optional): a file containing stochastic matrix entries
- Priority order file for master (optional): a file containing a branching order for the master branch-and-bound procedure
- Start information file (optional): a file containing start informations such as a feasible solution and/or a lower bound

A comfortable way to invoke the program on a Unix system is the command

```
ddsip < files2sip
```

where the file *files2sip* may contain the lines displayed in Figure 2.

```
model.lp.gz
ddsip.config
model.ord
rhs.sc
cost.sc
matrix.sc
order.dat
start.in
```

Abbildung 2: Input file

4 The model file

The model file contains the single-scenario model. Its format can be one of the formats readable by `CPLEX`, as e.g. the lp- or mps-format. The gzipped forms of these files (with extensions .lp.gz or .mps.gz) can be used for the sake of saving disk space. This is especially advisable when the parameter OUTFILE is chosen greater than 2.

The only requirement on the model file is that all first-stage variables are identified by a prefix or postfix appended to their names which is specified in the specs file.
There are no requirements on the ordering of the variables and constraints like in earlier `ddsip` versions.

5 The specification file

Comments could be included anywhere in this file apart from the CPLEX parameter section. An Asterisk identifies the beginning of a comment until the end of the line. This way explanations could be included and parameter lines temporarily inactivated. Commenting out whole lines could not be used within the CPLEX parameter section, but comments can be added after a parameter number/value pair in this section, too.
Keywords for parameters are accepted if their first six characters match the strings given in the tables. Unknown keywords are simply ignored. Only the lines with the **first occurrence** of each keyword are evaluated.
A sample specification file is given in Appendix A.

5.1 Parameters for the two-stage model

Hopefully, the identifiers in the first part of the file are self explaining. Otherwise, they should become clear when consulting the two-stage chapter of one of the standard text books on stochastic programming, see Birge and Louveaux (1997); Kall and Wallace (1994); Prekopa (1995).
The parameters FIRSTVAR, FIRSTCON, SECVAR, SECCON used in the older `ddsip` versions are now obsolete. These values and the corresponding indices of variables and constraints are now determined by the program using the prefix or postfix specified.
Either a prefix or a postfix of the variable names for identifying the first-stage variables has to be specified.

Name	Type	Default/Description	
POSTFIX	String	*none*	postfix for first-stage variables
PREFIX	String	*none*	prefix for first-stage variables
SCENAR	Int	0	Number of scenarios
STOCRHS	Int	0	Number of stochastic rhs elements
STOCCOST	Int	0	Number of stochastic cost coefficients
STOCMAT	Int	0	Number of stochastic matrix entries

Table 1: Problem specification parameters

5.2 CPLEX parameters

CPLEX parameters to be set different from their defaults have to be specified following a line with the indicator *CPLEXBEGIN*.
The contained lines have to start with the CPLEX parameter number followed

by the parameter value. This can be followed by a comment, starting with an asterisk. The parameter numbers are specified in the CPLEX callable library documentation, section parameter table.
The settings following directly the line with *CPLEXBEGIN* are applied to all problems solved.

Special settings for CPLEX parameters can be specified for the different stages of the program.

- For the calculation of the expected value problem (with all stochastic entries set to their expected values) parameter values can be overwritten after the identifier *CPLEXEEV*,
- for the evaluation of lower bounds after the identifier *CPLEXLB*,
- for a continuation of the evaluation of lower bounds with different settings after the identifier *CPLEX2LB*,
- for the evaluation of upper bounds after the identifier *CPLEXUB*,
- for a continuation of the evaluation of upper bounds with different settings after the identifier *CPLEX2UB*,
- for the subproblems of the Lagrangian dual after the identifier *CPLEX-DUAL*,
- for a continuation of the subproblems of the Lagrangian dual after the identifier *CPLEX2DUAL*.

For some problems it is beneficial to use two optimization calls with different parameter setings (e.g. mip emphasis, heuristic frequency) sequentially, the second one starting with the B&B tree and solutions obtained in the first call. This is enabled for lower and upper bound calculations and the Lagrangean dual if the sections beginning with *CPLEX2LB*, *CPLEX2UB*, or *CPLEX2DUAL* are specified.

The CPLEX parameter section has to end with the identifier *CPLEXEND*. Parameters not present in the mentioned chapters are set according to their CPLEX default values, cf. CPLEX (2016).
Note that in the section between *CPLEXBEGIN* and *CPLEXEND* there could be comments following the pair of numerical parameter identifier and value, but not whole lines could be comments, ie.e. no line should start with an asterisk.

Since at each node all scenario problems have to be solved for the dual bound and in every heuristic for finding/the improvement of the primal bound solving the single-scenario problems should be as fast as possible. So tuning the CPLEX paramaters (especially for scenarios difficult to solve) and specifying adapted parameter settings in the specs file is strongly encouraged, it can later on save a lot of solution time.
The tuning process is facilitated by setting the parameter OUTFIL to 4, CBFREQ to 0, NODELIM to 0, and HEURIST e.g. to 3. All the single-scenario problems for dual bounds (lb...) and primal bounds (ub...) are then written to the directory sipout for use in determining good CPLEX settings.

5.3 Parameters for the decomposition procedure

The parameters that effect the amount of output and the termination behavior are listed in the table below. The first column of the table contains the name of the parameter, the second one specifies whether it is an integer or a real (Dbl) parameter, and the last column explains the parameter's meaning.
The signal SIGTERM (as defined on LINUX/UNIX-environments) is handled by the program, other signals are not handled separately. SIGTERM causes the termination of `ddsip` after solving the current subproblem.
So far all termination parameters are static in the sense that they cannot be changed during the branch-and-bound algorithm. Therefore, a careful trade-off between these parameters and the termination parameters of the bundle method, see Section 5.6, is essential for the numerical performance, cf. Carøe and Schultz (1998).

Parameters that effect the behavior of the branch-and-bound algorithm are compiled in the Tables 2 and 3. The default values are marked with a star. The use of start values is described in Section 7.1, the use of priority order information in Section 7.2. The parameter RELAXL effects the relaxation levels during the evaluation of lower bounds. It relates to the scenario subproblems.
The two types of the deterministic equivalent specified by DETEQT differ only in the objective. In both cases additional variables with names DDSIP_objSC001 etc. are added plus equations assigning the corresponding scenario objective values. In the type 0 equivalent the objective is the sum of these additional variables multiplied by the scenario probability. In the type 1 file the objective is formed using the original first-stage variables.
The *branching value* is used to create two new subproblems as described in *STEP 5* of the decomposition algorithm. For continuous components it depends on the parameter EPSILON.
The dispersion norm of a node is calculated as $\max_j\{\max_i x_{ij} - \min_i x_{ij}\}$ where x_j, $j = 1, \ldots, S$, are the (first-stage part of the) solutions of the S subproblems and x_{ij}, $i = 1, \ldots, n$, are their i-th components. Nodes with a dispersion norm smaller than NULLDISP are considered as leaves of the B&B tree (i.e. no further branching on them).

Name	Type	Range	Default	Description
OUTLEV	Int	0..100	0	Amount of output to *more.out*. **Caution:** The file *more.out* may become large for high values of OUTLEV!
OUTFIL	Int	0..6	1	Amount of output files, see chapter 8. **Caution:** If OUTFIL is greater than 3, lp- or sav-files are written at each node and for each scenario!
LOGFRE	Int	0..	1	A line of output is printed every i-th iteration.
NODELI	Int	0..	10000	The node limit for the branch-and-bound procedure.
TIMELI	Dbl	0..	86400.	The total time limit in seconds (CPU-time) including the time needed to solve the EEV problem.
ABSOLU	Dbl	0..	0	The absolute duality gap.
RELATI	Dbl	1e-10..	1e-4	The relative duality gap.
EEVPRO	Int	0..1	0	If the parameter is 1, then solve the EEV-problem and report the VSS, cf. Birge and Louveaux (1997).
DETEQU	Int	0..1	0	If the parameter is 1, then write a deterministic equivalent to sipout/det_equ.lp.gz Only for expectation-based risk models 0, 1, 2, 4.
DETEQT	Int	0..1	0	Determines the form of the obj. of the deterministic equivalent.

Table 2: Output and termination parameters

The two types of the deterministic equivalent specified by DETEQT differ only in the objective. In both cases additional variables with names DDSIP_objSC001 etc. are added plus equations assigning the corresponding scenario objective values. In the type 0 equivalent the objective is the sum of these additional variables multiplied by the scenario probability. In the type 1 file the objective is formed using the original first-stage variables.
The *branching value* is used to create two new subproblems as described in *STEP 5* of the decomposition algorithm. For continuous components it depends on the parameter EPSILON.
The dispersion norm of a node is calculated as $\max_j\{\max_i x_{ij} - \min_i x_{ij}\}$ where x_j, $j = 1, \dots, S$, are the (first-stage part of the) solutions of the S subproblems and x_{ij}, $i = 1, \dots, n$, are their i-th components. Nodes with a dispersion norm smaller than NULLDISP are considered as leaves of the B&B tree (i.e. no further branching on them).

Name	Type	Range	Default	Description
ACCURA	Dbl	1e-13..1	1e-12	Accuracy – real values are considered equal if the absolute value of their difference is less than accuracy
EPSILO	Dbl	1e.10..	1e-10	Specifies the gap between disjoint subdomains if continuous variables are branched.
NULLDI	Dbl	5e-10..	5e-10	Branch nodes only if their dispersion norm is greater than NULLDI.
CPXORD	Int	0..1	0	Read a CPLEX priority order?
PORDER	Int	0..1	0	Use priority order for branching? If this parameter is 1, a priority order file has to be specified.
STARTI	Int	0..1	0	Use start information? If this is 1, a file containing the start values and/or lower bound has to be specified.
MAXINH	Int	0..	5	Maximal level of "inheritance" of scen. solutions in the B&B-tree.
HOTSTA	Int	0..6	0	No warm starts during B&B.
			1*	Use solution pool of previous scenario and integer values of the same scenario in father as initial solutions.
			2	Use solution pool of previous scenario and lower bound from father node (not applicable for risk models).
			3	Use solutions of all previous scenarios in the same node.
			4	Use solutions of all previous scenarios of the same node as well as solutions of all scenarios in father node.
			5	Uses solution info as in 3 together with the lower bound from father node.
			6	Uses solution info as in 4 together with the lower bound from father node.
CBHOTS	Int	0..4	0	No warm starts/reordering of scenarios within ConicBundle.

continued on next page

continued from previous page

Name	Type	Range	Default/Description	
			1*	Use scenario solutions as MIP starts in the root node for the first 3 descent steps.
			2*	Use scenario solutions as MIP starts in the root node for the all descent steps.
			3*	Use scenario solutions as MIP starts in the root node for the all descent steps.
			4*	Use scenario solutions as MIP starts in all nodes for the all descent steps.
BRADIR	Int	-1*,1	-1	Branching down is done first.
			1	Branching up is done first.
BRASTR	Int	0..2	0	Use the middle between maximal and minimal value for the chosen variable as branching value.
			1	Use the expected value (weighted average) of the scenario solutions for the chosen variable as branching value.
			2*	Use a point in between the weighted average of the scenario solutions for the chosen variable and the mean between maximal and minimal value for that variable as branching value.
BRAEQU	Int	-1..1	1	"equal distribution" – Among the variables with maximal dispersion value choose one for branching which most equally divides feasibility of scenario solutions among both new nodes
			0*	"mixed" – in 3/5 of the iterations choose a variable such that most of the scenario solutions remain feasible for one of the new nodes, in the others equally distributed
			-1	"unequally" – in all iterations choose a variable such that most of the scenario solutions remain feasible for one of the new nodes

continued on next page

continued from previous page

Name	Type	Range	Default/Description	
INTFIR	Int	0..1	0	Branching exclusively according to branching rules.
			1*	Branching on integers first.
BOUSTR	Int	0..10	0	choose node with best bound as next node
			1	heuristic: among nodes with sufficiently small bound choose node with biggest dispersion norm (unsolved nodes first)
			2	heuristic: among nodes with sufficiently small bound choose node with biggest dispersion norm
			3	heuristic: among nodes with sufficiently small bound choose node with least number of violations (unsolved nodes first)
			4	heuristic: among nodes with sufficiently small bound choose node with least number of violations
			5-9	heuristic: depth first until a incumbent is found, then switch to BOUSTRAT-5
			10*	heuristic: among last four generated nodes choose node with best bound (unsolved nodes first) as long as its objective value does not exceed a threshold. Choose the best bound node otherwise and also every 25th node.
BESTFR	Int	1..	24	frequency for choosing the best bound node regardless of the other criteria
BTTOLE	Dbl	1e-4..1	0.2	(with BOUSTR 10): tolerance for using the best bound node in the next 4 nodes
PERIOD	Int	1..	32	for BOUSTRAT>0: determines the period of bounding steps with changing tolerances for "sufficiently small"

continued on next page

continued from previous page

Name	Type	Range	Default	Description
TOLSMA	Int	1..PERIOD	16	for BOUSTRAT>0: determines the number of steps in each period with a small tolerance
KAPPA	Int	0..2	0*	No gathering of kappa values by CPLEX
	Int		1	sample gathering of kappa values by CPLEX and reporting
	Int		2	full gathering of kappa values by CPLEX and reporting
RELAXL	Int	0..2	0*	No integrality relaxation.
			1	Relax first-stage variables.
			2	Relax first- and second-stage variables.
QUANTI	Int	0..	10	Number of Quantiles to be displayed at the end
MAXINH	Int	0..	5	maximal level of inheritance of solutions in lower bounding
HEURIS	Int	1..14,99, 100	100	choice of heuristics, cf. Table 5
INTHEU	Int	-1..1		should the evaluation of different heuristics be interrupted when the relative gap is reached?
			0	do not interrupt
			1	interrupt instantly
			-1	continue through the list 1-10 if HEURIS is 100, but don't use 12 default: 0 for HEURIS 99, -1 for HEURIS 100
ADDBEN	Int	0..1	1	Should Benders feasibility cuts be added?
TESTBE	Int	0..1	1	Should be tested for further cuts from the remaining scenarios if one cut is violated?
REINIT	Int	0..	10	In the root node: how many times should the first lower bounding/upper bounding be repeated? (only as long as a new cut was added, which increases the bound)

continued on next page

continued from previous page

Name	Type	Range	Default/Description	
ADDINT	Int	0..1	1	For pure binary first stage: should integer cuts be added excluding the current vertex in case of infeasibility?
PREMA	Int	0..1	1	Should premature stop in upper bounding be used?

Table 3: Branch-and-bound parameters

The MAXINHERIT parameter specifies a maximal level of passing on to the descendant nodes solutions and bounds in the B&B-tree for the lower bound step. This facility is interrupted by an invocation of the conic bundle algorithm, which will change the Lagrangian multipliers. If a Benders feasibility cut is added the feasibility of single scenario solutions of the node being branched on is checked with regard to the added cuts. Inheritance of solutions saves up to half of the evaluations of scenario problems, but in case these problems are not solved to optimality the lower bound might be inferior to the one possibly obtained for the descendants (with the addtional bounds on the variables branched on). For this reason the level of inheritance can be bounded to the number of branching steps specified with the MAXINHERIT keyword.
After the first node the scenarios are sorted descending wrt. their contribution to the expected value. This serves earlier detection of inferiority and allows stopping of evaluation of scenarios.
If conic bundle is used, the lower bound for each scenario is not fully reliable in upper bounding (which is done without Lagrangean parameters). The parameter PREMATURE (default 1) to allow/disallow premature stopping in upper bounding can be set to 0 when conic bundle is used.

5.4 Heuristic

The decomposition algorithm uses a number of heuristic to guess feasible first-stage solutions based on the solutions of the scenario subproblems in the current node. The heuristic is set by means of the parameter HEURIS. The possible values of HEURIS are listed in the following table.
More than one of the heuristics could be used sequentially in every node when specifying the value 99, followed by a list of heuristics to be used, the specs file line could look like this:

HEURISTIC 99 2 3 12 * heuristics 2, 3, **and** 12 are specified

The default is the value 100, which applies all heuristics 1 to 11 in every node plus heuristic 12 (test all scenario solutions) in the first 11 nodes, in other nodes under certain conditions, and every 200th node.

Value	Description
1	The average of the solutions is used. Integer components are rounded down.
2	The average of the solutions is used. Integer components are rounded up.
3	The average of the solutions is used. Integer components are rounded to the nearest integer.
4	The solution occurring most frequently is used.
5	Use the solution of a scenario that is closest (l_1 norm) to average.
6	The first-stage solution with highest probability is chosen (adding probabilities in case of identical first-stage scenario solutions)
7	Solution with best objective value
8	Solution with worst objective value
9	Solution with minimal sum of first-stage variables
10	Solution with maximal sum of first-stage variables
11	Try solutions of those scenarios which occur more than scenarios/5 times
12	Try solutions of all scenarios
13	Apply heuristic 4 and 5 alternating
14	Apply heuristic 3 and 5 alternating
99	Indicates that a list of heuristics is given in the sequel. See the example in the Appendix.
100*	Heuristics 1-11 are applied plus the more expensive heuristic 12, if one of the others found a new incumbent. Heuristic 12 is also applied in the first 11 nodes to facilitate dicovery of an incumbent early in the B&B tree.

Table 5: Values of parameter HEURIS

In order to save time the order of evaluation of the scenario problems is changed if an infeasible one is encountered. Since often the same scenarios give infeasibility these ones are evaluated first in the sequel.

5.5 Parameters for the risk model

The parameters for the risk models are displayed below. Some parameter settings lead to ill-posed models, cf. Märkert (2004). `ddsip` provides only limited consistency checks with this respect.
The same holds true for the choice of an algorithm. Algorithms similar to the scenario decomposition as outlined above only apply to mean-risk models with *decomposable* linear risk measures. The algorithm FSD can be used with any risk measure that is consistent with first-order stochastic dominance. The weakest

algorithm NFSD allows the minimization of any (nonlinear) risk measure. A detailed description of the different algorithms can be found in Märkert (2004).

Name	Type	Range	Default	Description
RISKMO	Int	-7..7	0	Risk model
WEIGHT	Dbl	0..	0.9	Weight on risk term in objective.
TARGET	Dbl	..	-1e+20	Target for target measures.
PROBLE	Dbl	0..1	0	Probability level for (T)VaR.
RISKAL	Int	0..2	0*	Scenario decomposition.
			1	FSD-algorithm (advised for TVaR, also for pure risk models except 4 (worst case costs) and 7 (stadard deviation)).
			2	NFSD-algorithm.
BRAETA	Int	0..1	1	Branch order of additional first-stage variable in models 4 and 5. A value of 0 means that this variable will not be branched and its dispersion ignored.
RISKBM	Dbl	..	1e+20	Big M for risk model (-)2, also used as bound for η with risk model 5.

Table 6: Parameters for the risk model

The program offers 7 different mean-risk models and the associated risk models, in which the expected value is not minimized. In the following table we have compiled the possible settings.

Name	Value	Description
RISKMO	$1, -1$	Expected excess above target.
	$2, -2$	Excess probabilities.
	$3, -3$	Absolute semideviation.
	$4, -4$	Worst-case-costs.
	$5, -5$	Tail value-at-risk.
	$6, -6$	Value-at-risk.
	$7, -7$	Standard deviation

Table 7: Risk models

Positive values of RISKMO implement the model $\min_X \mathbb{E}X + \alpha\mathbb{R}X$, negative ones the risk model $\min_X \mathbb{R}X$. These pure risk models are often harder to solve.

Pure risk models do not work at the moment, these options are now disabled.

For the (mean-)absolute semideviation model the TARGET has to be set less or equal the optimal expected value when using RISKALG 0 (cf. Märkert (2004)).

5.6 Parameters for the dual method

There is an online manual of `ConicBundle` available Helmberg (2012). We refer also to Helmberg (2000) where the method is described in the context of semidefinite programming. Below, we only explain the parameters defined in our implementation.
The dual method cannot be used in combination with the algorithm based on the FSD-consistency of the risk measure, cf. Märkert (2004). The user has to take care about this detail.
The use of stochastic cost coefficient in combination with the Lagrangian dual is not supported, yet. We recommend to reformulate the problem by an additional variable and an additional constraint representing the objective function. This leads to a stochastic matrix.
The ConicBundle weight is changed between single dual steps under certain conditions by ConicBundle itself. `ddsip` offers the possibility to heuristically change (increase or decrease) the weight between descent steps. By default this feature is turned on and could be deactivated by parameters.

Name	Type	Range	Default/Description	
CBFREQ	Int	..	-16	Use `ConicBundle` in every ith node. Negative numbers are interpreted as —i—th node plus preceding one.
CBITLI	Int	0..	25	Iteration limit: descent steps.
CBRITLI	Int	0..	CBITLIM+7	Iteration limit in root node: descent steps.
CBSTEP	Int	0..	12	Maximal number of iterations within one descent step.
CBTOTI	Int	0..	5000	Iteration limit: total iterations, i.e. descent and null steps.
CBCONT	Int	1..	6	Used when CBFREQ<0: number of nodes with invocation of `ConicBundle` in every node after CBBREAK.
CBBREA	Int	1..	24	Used when CBFREQ<0: invocation of `ConicBundle` in nodes 1..CBBREAK only for nodes, which inherited more than half of the scenario solutions.
CBPREC	Dbl	0..	1e-14	Precision of bundle method.
CBPRIN	Int	0..	0	`ConicBundle` output level.
NONANT	Int	1..3		The identity of the first-stage vectors is represented by
			1*	$x_1 = x_2, x_1 = x_3, \dots, x_1 = x_n$
			2	$x_1 = x_2, x_2 = x_3, \dots, x_{n-1} = x_n$
			3	$p_i x_i = \sum_{j=1, j \neq i}^{n} p_j x_j \ \forall i$
CBBUNS	Int	1..	500	Maximal bundle size
CBWEIG	Dbl	0..	1.	Initial weight (cf. CB manual)
CBFACT	Dbl	0..1	0.01	Initial weight in the next node is final (1-CBFACTOR)*(weight in the father) + CBFACTOR*(initial weight)
CBINHE	Int	0,1	0	Should the solutions of the father node be inherited in the initial evaluation?
CBCHAN	Int	0,1	0	Should the tolerances and time limits for CPLEX be temporarily changed after a number of unsuccessful steps?
CBREDU	Int	0,1	1	Should the weight be decreased after successful steps with no or few null steps?
CBINCR	Int	0,1	1	Should the weight be increased after steps with too many null steps?

Table 8: `ConicBundle` parameters

6 The scenario files

Unfortunately, the program `ddsip` is not conform to the SMPS data format, yet. But in contrast to the SMPS format due to the requirements for the variable and constraint names and the input format for the stochstic coefficients no specific ordering is required in the model file. The proprietary data format of the scenario files is described below.
Each file has to start with the keyword 'Names' followed by a list of the names of constraints for the stochastic right-hand sides, variables for the stochastic objective coefficients, and pairs of names for constraint and variable for the stochastic matrix entries. These name lists determine the order of objects the coefficients in the different scenarios are assigned.
The identifier *sce* indicates the begin of the entries for the next scenario. Figure 3 displays the different scenario files.

right-hand sides	cost coefficients	matrix entries
Names	Names	Names
constr25	var16	cons5 var15
constr30	var18	cons5 var16
⋮	⋮	⋮
scenario1	scenario1	scenario1
0.1	23.1	0.125
12.12	33.33	34
32.4		
⋮	⋮	⋮
scenarioN	scenarioN	scenarioN
0.1	25.56	30.3
30.2	11.5	22.56
41.0		
⋮	⋮	⋮

Abbildung 3: Format of the scenario files

In the right-hand side scenario file, the first number after the identifier is the scenario probability. For problems without stochastic right-hand sides, this file has to contain only the probabilities of the individual scenarios (and does not contain the 'Names' keyword, but does have the *sce* keyword for each scenario followed on the next line by the probability).
In the cost coefficient scenario file the probability entries are left out. Otherwise the entries are the same as those in the right-hand side scenario file, cf. Figure 3. The number of stochastic cost coefficients c has to be specified in the specification file.

7 Optional files

7.1 Start information file

Start information can be provided as displayed in Figure 4. Hereby, BEST

```
BEST          2345
BOUND         2000
SOLUTION
1
2
⋮
MULTIPLIER
3
4
⋮
```

Abbildung 4: File with start values

should be an upper bound, BOUND a lower bound, SOLUTION a feasible first-stage solution, and MULTIPLIER a Lagrangian multiplier. Any item of these four different informations may be left out.
The user has to care for the consistency of the data.

7.2 The priority order files

If indicated by setting the CPLEX parameter CPX_PARAM_MIPORDIND, the name of a CPLEX branching order file to be used for the scenario problems has to be specified following the name of the model file (named model.ord in Figure 2). Its format is described in the CPLEX manual.
If the parameter PORDER is set, a file with branching order information for the master problem has to be given (named order.dat in Figure 2). The file has two columns whereby the first one contains **names** of first-stage variables and the second one integer values greater than 1 (1 is the default, variable with default priority need not be specified). High values lead to early branching.

8 Output

8.1 Output on screen

The following shows typical lines of output as produced by `ddsip` (accuracy of the values cut due to line length). The meaning of the single entries is rather straightforward. Here is a short explanation.

Node	gives the number of the currently solved node.
Nodes	counts the number of nodes generated in the tree.
Left	counts the number of nodes in the front tree.

```
 Node Nodes Left  Objective        Heuristic  Best Value    Bound     Viol./Dispersion    Gap Wall Time CPU Time Father█
    0     1    1  44841.3565      infeasible               44841.356    5  3                   0h 00:03 0h 00:03  -1
    1     3    2  47944.6273      infeasible               44841.356    7  2.7888              0h 00:06 0h 00:06   0
    2     3    2  44844.5722      infeasible               44844.572    5  2.7888              0h 00:07 0h 00:07   0
    3     5    3  44953.0125      infeasible               44844.572    7  2.7888              0h 00:09 0h 00:08   2
    4     5    3  44848.1886        multiple               44848.188    4  2.7888              0h 00:09 0h 00:09   2
    5     7    4  45549.6459      infeasible               44848.188    5  2.7888              0h 00:10 0h 00:10   4
    6     7    4  44854.2998      infeasible               44854.299    3  2.7888              0h 00:12 0h 00:12   4
*   7     9    5  Heuristic  2   66346.70463 66346.70463                                      0h 00:14 0h 00:13
    7     9    5  45712.4035     66346.70463 66346.70463   44854.299    6  2.7888      32.39%  0h 00:14 0h 00:13   6
*   8     9    5  Heuristic  2   44864.77749 44864.77749                                      0h 00:16 0h 00:16
    8     9    1  44859.2107     44864.77749 44864.77749   44859.210    2  2.7888     0.0124%  0h 00:16 0h 00:16   6
    9    11    1    cutoff                   44864.77749   44859.210                 0.0124%  0h 00:17 0h 00:17   8
   10    11    1  44863.2847     44865.25242 44864.77749   44863.284    1  2.7888    0.00332%  0h 00:20 0h 00:19   8
   11    13    2  44864.4325     44865.05453 44864.77749   44863.284    1  1.162     0.00332%  0h 00:22 0h 00:21  10
*  12    13    2  Heuristic 12   44864.65876 44864.65876                                      0h 00:24 0h 00:23
*  12    13    2  Heuristic  3   44864.38172 44864.38172                                      0h 00:25 0h 00:24
   12    13    1  44863.5109     44864.38172 44864.38172   44863.510    1  1.6268    0.00194%  0h 00:25 0h 00:24  10
   13    15    2  44864.1805     44864.54332 44864.38172   44863.510    1  0.67783   0.00194%  0h 00:27 0h 00:26  12
```

Objective	is the lower bound of the current node.
Heuristic	is the upper bound returned by the heuristic.
Best Value	is the overall upper bound.
Bound	is the overall lower bound.
Viol.	is the number of variables violationg the nonanticipativity.
Dispersion	is the greatest difference of first-stage variable values.
Gap	is the relative gap between *Best Value* and *Bound*.
Wall Time	gives the wall time passed since invoking the program.
CPU Time	gives the CPU time passed since invoking the program.
Father	gives the number of the father node.

The pruning of nodes is indicated by the entry *cutoff* in the row 'Objective'. The row 'Heuristic' may also contain the entries *n-stop* (inferiority, evaluation stopped at n-th scenario) or *multiple* (a solution has been evaluated previously). When more than one solution is evaluated in a branch-and-bound step, the entry indicates the status of the last solution.
An asterisk in the first position indicates that a new best value was found. In the example a list of heuristics is used in every node, the lines beginning with an asterisk indicate which heuristic found a new best solution. If the scenario solutions of a node fulfil the nonanticipativiy and yield a new best upper bound, in the output line for the node an asterisk is prepended, too.

8.2 Output in the file *sip.out*

All output files will be placed in the subdirectory *sipout* of the current directory. This directory will be created if it does not exist. A further run of `ddsip` overwrites the existing output files. The output on screen is also directed to the file *more.out* in case OUTLEV > 51. In addition, this file contains the parameters read and some information on the solution:

- The value of *Status* indicates the solution status:

-1	the process was terminated by the user,
1	the node limit has been reached,
2	the gap (absolute or relative) has been reached,
3	the time limit has been reached,
4	the maximal dispersion, i.e. the maximal difference of the first-stage components within all remaining front nodes, was less then the parameter NULLDISP (null dispersion),
5	the whole branching tree was backtracked.

- *Time* is the total CPU time needed by `ddsip`.
- *Upper bounds* is the number of evaluated upper bounds.
- *Tree depth* is the depth of the branch-and-bound tree.
- *Nodes* is the total number of nodes.
- *EEV* is the solution of the EEV problem according to Birge and Louveaux (1997).
- *VSS* is the value of stochastic programming according to Birge and Louveaux (1997).
- *Expectedvalue* is the expected value.
- *Riskmeasure* is the value of the used risk measure.

8.3 Other output files

The number of additional output files and their content is ruled via the parameters OUTFIL and OUTLEV. Values of OUTFIL greater than 1 are intended for debugging purposes.

- The file *more.out* contains more or less information depending on the parameter OUTLEV, e.g. the subproblem solutions, the branch-and-bound tree, and the objective function composition.
- The file *solution.out* contains information on the solution with the optimal first-stage solution among them. It is not written when OUTFILES is set to 0.
- If OUTFILES is greater than 1, then in addition the following files are written to sipout:
  - The files *rhs.out*, *cost.out*, *matrix.out*, and *model.lp.gz* offer a check for a correct reading of the scenario files and the model file, respectively.
  - The files *risk.lp.gz* and *eev.lp.gz* document the changes made during defining the risk model and solving the expected value problem, respectively.

- If OUTFILES equals 3, then in addition the following files are written to sipout:
  - The single scenario files *lb_sc_*_n0_gc*.lp.gz* and *ub_sc_*_n0_gc*.lp.gz* solved in the root node for lower and upper bounds, respectively.
- If OUTFILES equals 4, then in addition the following files are written to sipout:
  - The single scenario files *lb_sc_*_n*_gc*.lp.gz* and *ub_sc_*_n*_gc*.lp.gz* solved in all nodes for lower and upper bounds.
- If OUTFILES equals 5, then the files *lb_sc_*_n*_gc*.lp.gz* and *ub_sc_*_n*_gc*.lp.gz* are not written, but instead
  - The single scenario files *lb_sc_*_n*.sav.gz* solved in all nodes for lower bounds plus the used MIP starts as *lb_sc_*_n*.mst.gz* and the CPLEX settings as *lb_sc_*_n*_1.prm* and *lb_sc_*_n*_2.prm*. In the case of conic bundle use these are overwritten in every dual step, just the ones of the final dual step of each node (or the one causing a program crash) remain.
- If OUTFILES equals 6, as in the case 5 the single scenario problem files are not written as lp.gz, but instead
  - The single scenario files *lb_sc_*_n*_gc*.sav.gz* solved in all nodes for lower bounds plus the used MIP starts as *lb_sc_*_n*_gc*.mst.gz* and the CPLEX settings as *lb_sc_*_n*_1.prm* and *lb_sc_*_n*_2.prm*. This is the same amount of output files as with the setting 5 in case no conic bundle is used. In the case of conic bundle use these are written in every dual step, increasing the value after gc (global counter) with each step.

In order to facilitate debugging/tuning of parameters two possibilities have been included in order to get information in selected stages of the run without producing a huge amount of output files for the overall run:
issuing the command 'kill -SIGUSR1 process_number' once switches the OUTFILE parameter to 6, issuing the same kill command again switches it back. Analogously issuing the command 'kill -SIGUSR2 process_number' once switches the CPLEX iteration log to on, issuing the same kill command again switches it back off.

9 License and bugs

9.1 License

The program is free software. It is distributed under the GNU General Public License as published by the Free Software Foundation, see GNU Project (2004).

9.2 Bug report

Please report all bugs via email to ralf.gollmer@uni-due.de. If possible, provide the input files to facilitate reproduction of the error.

A Sample specs files

```
BEGIN
* Parameters to specify the two-stage model

PREFIX          S1_   * Prefix of the names of first-stage variables
STOCRHS          70   * Number of stochastic rhs elements
SCENARIOS         5   * Number of scenarios

STOCCOST          0   * Number of stochastic cost coefficients
                      * (stoch. obj. doesn't work together with risk!)
STOCMAT           0   * Number of stochastic matrix entries

* Parameters for dual decomposition procedure

OUTLEVEL          5   * Print info to more.out
NODELIM          90   * ddsip node limit
TIMELIM         900   * ddsip time limit in seconds
HEURISTIC    99 3 4   * Heuristics to produce a feasible solution
EEVPROB           1   * Solve EEV, cf. Birge and Louveaux (1997)
ABSOLUTE          0   * Absolute duality gap
RELATIVE      0.001   * Relative duality gap

BRADIR           -1   * Branching direction
BRASTRAT          0   * Branching strategy
BOUSTRAT          0   * Bounding strategy
EPSILON        1e-9   * Epsilon used for branching on real variables
RELAXL            1   * Relaxation level (1= relax first-stage variables
                      * in scenario problems)
LOGFREQ           1   * Output log frequency
INTFIRST          1   * Branching on integers first
ACCURACY      1e-12   * Accuracy used to compare real values
NULLDISP      1e-12   * Tolerance level for null-dispersion
                      * (used together with ACCURACY!)
QUANTILES        10   * Number of Quantiles


* Risk model

RISKMOD           1   * Risk model
TARGET         1e+3   * Target for target measures
WEIGHT         1e+0   * Weight of the risk term

RISKBM         1e+4   * Big M used in modeling risk models 2 and 5
BRAETA            0   * Branch on auxiliary variable eta?
                      * (risk models 4 and 5)
* CPLEX Parameters (See CPLEX manual CPLEX (2016))

CPLEXBEGIN
1035              1   * Output on screen indicator
2009           1e-7   * Relative gap
CPLEXEEV
2009            0.1   * Relative gap for EEV
CPLEXLB
2058              3   * MIP emphasis
2031             10   * heuristic frequency
2009          0.002   * Relative Gap
1039            900   * Time limit
CPLEX2LB
2058              4   * MIP emphasis
2031              5   * heuristic frequency
2061             20   * RINS neighbourhood search heuristic frequency
2009           1e-6   * Relative Gap
1039            350   * Time limit
CPLEXUB
2058              3   * MIP emphasis
2031             10   * heuristic frequency
1039            750   * Time limit
2009          0.001   * Relative Gap
CPLEX2UB
2058              4   * MIP emphasis
2031             15   * heuristic frequency
2061             60   * RINS neighbourhood search heuristic frequency
1039            300   * Time limit
CPLEXDUAL
2009           1e-4   * Relative gap for Lagrangian Dual
1035              0   * Output on screen indicator for Lagrangian dual
CPLEXEND

* Parameters specifying the use of ConicBundle

CBFREQ        -1000   * Conic Bundle in every i-th node
CBITLIM          20   * Limit for number of descent steps in CB
CBTOTIT        1000   * Limit for total number of CB iterations
                      * (incl. null steps)
NONANT            2   * Nonanticipativity representation
CBPRINT           1   * Output level of Conic Bundle
CBBUNS          200   * Maximal bundle size
CBWEIGHT         1.   * Conic Bundle initial weight
END
```

Specification file 1

Another example using mostly default parameter settings:

```
BEGIN
POSTFIX          _FIRSTSTAGE
SCENAR                    10
STOCRHS                 1000
STOCCOST                   0
STOCMAT                   31

CPLEXBEGIN
1067                       1  * number of threads
1035                       0  * Output on screen indicator
2008                     0.0  * Absolute Gap
2009                   1.e-8  * Relative Gap
1039                    1200  * Time limit
1016                    1e-9  * simplex feasibility tolerance
1014                    1e-9  * simplex optimality tolerance
1065                   40000  * Memory for working storage
2010                   1e-20  * integrality tolerance
2008                       0  * Absolute gap
2020                       0  * Priority order
2053                       2  * disjunctive cuts
2040                       2  * flow cover cuts
2049                       1  * Gomory cuts
2052                       2  * MIRcuts
2064                       0  * repeat presolve
CPLEXDUAL
2009                   1.e-1  * Relative Gap
2058                       0  * MIP emphasis
2098                       2  * feasibility pump
2031                       3  * Heuristic frequency
CPLEX2DUAL
2009                   1.e-7  * Relative Gap
2058                       4  * MIP emphasis
CPLEXEND

OUTLEVEL                   5  * Print info to more.out

TIMELIMI              345600  * Time limit 96h
CBITLIM                   15  * Number of CB descent steps
                              * (default in root node: CBITLIM+7)
RELATIVEGAP             1e-5  * Relative duality gap

END
```

Literatur

Ahmed, S.; M. Tawarmalani; N. V. Sahinidis (2000), A Finite Branch and Bound Algorithm for Two-Stage Stochastic Integer Programs, Stochastic Programming E-Print Series, `http://www.speps.info`

Beale, E.M.L. (1955), On Minimizing a Convex Function Subject to Linear Inequalities, Journal of Royal Statistical Society 17, pp. 173-184.

Birge, J.R.; F.V. Louveaux (1997), Introduction to Stochastic Programming, Springer, New York.

Carøe, C.C.; R. Schultz (1999), Dual Decomposition in Stochastic Integer Programming, Operations Research Letters 24, pp. 37-45.

Carøe, C.C.; R. Schultz (1998), A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 98-11.

Dantzig, G.B. (1955), Linear Programming under Uncertainty, Management Science 1, pp. 197-206.

GNU Project (2004), `http://www.gnu.org`

C. Helmberg (2000), Semidefinite Programming for Combinatorial Optimization, ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin.

C. Helmberg (2012), `http://www-user.tu-chemnitz.de/~helmberg/ConicBundle/`

IBM ILOG (2016), ILOG CPLEX 12.6.3 Reference Manual, installed with the product or online:
`http://www-01.ibm.com/support/docview.wss?uid=swg21503602`

Kall, P.; S.W. Wallace (1994), Stochastic Programming, Wiley, Chichester.

Laporte, G.; F.V. Louveaux (1993), The Integer L-Shape Method for Stochastic Integer Programs with Complete Recourse, Operations Research Letters 13, pp. 133-142.

Louveaux, F.V.; R. Schultz (2003), Stochastic Integer Programming, In: A. Ruszczynski, A. Shapiro (Eds.), Stochastic Programming, Elsevier, North-Holland.

A. Märkert (2004), Deviation Measures in Stochastic Programming with Mixed-Integer Recourse, Doctoral thesis, Institute of Mathematics, University Duisburg-Essen, Campus Duisburg.

Ogryszak, W.; A. Ruszczinsky (1999), From Stochastic Dominance to Mean-Risk Models: Semideviations as Risk Measures, European Journal of Operations Research 116, 33-50.

Prekopa, A. (1995), Stochastic Programming, Kluwer, Dordrecht.

Schultz, R.; S. Tiedemann (2003), Risk Aversion via Excess Probabilities in Stochastic Programs with Mixed-Integer Recourse, SIAM Journal on Optimization 14, pp. 115-138.

van der Vlerk, M.H. (1995), Stochastic Programming with Integer Recourse, PhD thesis, University of Groningen, The Netherlands.

Walkup, D.; R.J.-B. Wets (1967), Stochastic Programs with Recourse, SIAM Journal on Applied Mathematics 15, pp. 1299-1314.