

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 Duisburg
Forsthausweg 2, D-47048, Germany
gollmer@math.uni-duisburg.de

January 25, 2008

1 Introduction

`ddsip` is a C-implementation of a number of scenario decomposition algorithms for two-stage stochastic linear programs with mixed-integer recourse. The program is based on a previous Fortran 90-implementation of C.C. Carøe. Main idea of the decomposition algorithms is the Lagrangian relaxation of the non-anticipativity constraints and a branch-and-bound algorithm to reestablish non-anticipativity. 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 (2004). We use the CPLEX callable library to solve the mixed-integer subproblems in the branch-and-bound tree, see CPLEX (2002). The current version of `ddsip` relies on CPLEX 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 Ruszczyński (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 ask the user to support fixing bugs by reporting them to us as they occur.

We did not yet include support for the SMPS format, but a rather basic input format is implemented, which is not very handy with stochastic matrix entries. 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

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

The random parameter $\xi := (c, q, T, W, h) : \Omega \rightarrow \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{s \times n} \times \mathbb{R}^{s \times m} \times \mathbb{R}^s$ 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

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

where

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

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 $\mathcal{R} : \mathcal{Z} \rightarrow \mathbb{R}$, e.g. the expected value, some measure of risk, or a weighted sum of both, can serve as objective criterion

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

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.

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

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 , $j = 1, \dots, S$, with corresponding probabilities π_j . Then, problem (4) turns into

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

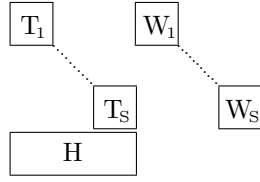


Figure 1: Constraints of (6)

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

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

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

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

where $M_j = \{(x_j, y_j) : T_j x_j + W_j y_j = h_j, x_j \in X, y_j \in Y\}$, $j = 1, \dots, S$.

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 $H = (H_1, \dots, H_S)$. The problem decomposes when we relax the non-anticipativity 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

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

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 $z^* = Q_E(x^*)$ 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 $z_{LD}(P)$ equals infinity (infeasibility of a subproblem) go to *STEP 2*.

STEP 4 Bounding: If $z_{LD}(P)$ 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 $z^* := \min\{z^*, Q_E(x_j)\}$, delete all $P' \in \mathcal{P}$ with $z_{LD}(P') \geq z^*$ and go to *STEP 2*.
- not identical, then compute a suggestion $\hat{x} := Heu(x_1, \dots, x_S)$ using some heuristic. Set $z^* := \min\{z^*, Q_E(\hat{x})\}$ and delete all $P' \in \mathcal{P}$ with $z_{LD}(P') \geq z^*$.

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.

- Specification file: a file containing the specifications of the stochastic program, some CPLEX- and b&b-parameters
- Model file: a file readable by CPLEX (e.g. lp- or mps-format, possibly in gzipped form) that specifies the single-scenario model
- 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
- 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.

sip.in model.lp model.ord rhs.sc cost.sc matrix.sc order.dat start.in
--

Figure 2: Input file

4 The specification file

Comments could be included anywhere in this file. 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 (not to be used in the CPLEX parameter section).

Keywords are accepted if their start matches the strings given in the tables. Unknown keywords are simply ignored. Lines with the first occurrence of each keyword are evaluated.

A sample specification file is given in Appendix A.

4.1 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). Some of the identifiers, as e.g. FIRSTVAR, are redundant. They serve to check consistency with the model file.

Either a prefix or a postfix for identifying the first-stage variables could be specified. The default is to expect a postfix of the form "01" in their names.

Name	Type	Default/Description
FIRSTC	Int	0 Number of First stage constraints
FIRSTV	Int	0 Number of First stage variables
SECCON	Int	0 Number of Second stage constraints
SECVAR	Int	0 Number of Second stage variables
POSTFIX	String	01 postfix for first-stage variables
PREFIX	String	<i>none</i> 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

4.2 CPLEX parameters

CPLEX parameters 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. The parameter numbers are specified in the CPLEX callable library documentation, section parameter table.

Special settings for CPLEX parameters can be specified for the different stages of the program. For the calculation of the EEV the parameter values can be overwritten after the identifier *CPLEXEEV*, for the evaluation of upper bounds after the identifier *CPLEXUB*, and for the subproblems of the Lagrangian dual after the identifier *CPLEXDUAL*. The CPLEX parameter section has to end with the identifier *CPLEXEND*. Parameters not present in this chapter are set according to their CPLEX default values, cf. CPLEX (2002).

4.3 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 4.6, 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.

Name	Type	Range	Default	Description—
OUTLEV	Int	0..90	0	Amount of output to <i>more.out</i> . Caution: The file <i>more.out</i> may become large for high values of OUTLEV!
OUTFIL	Int	0..5	0	Amount of output files, see chapter 8. Caution: If OUTFIL is greater than 3, lp-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..	1	The node limit for the branch-and-bound procedure.
TIMELI	Dbl	0..	100	The total time limit (CPU-time) including the time needed to solve the EEV problem.
ABSOLU	Dbl	0..	0	The absolute duality gap.
RELATI	Dbl	0..	0	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 produce a deterministic equivalent as <i>det_equ.lp.gz</i> . Only for expectation based problems (and it takes quite a while!).

Table 2: Output and termination parameters

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.

Name	Type	Range	Default/Description
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 parameter is 1, a file containing the start values and/or lower bound has to be specified.
MAXINH	Int	0..	10 Maximal level of "inheritance" of scen. solutions in the B&B-tree.
HOTSTA	Int	0..4	0* No warm starts during branch-and-bound.
			1 Use solution pool of previous scenario as initial solution.
			2 Use solution pool of previous scenario and lower bound of 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.
BRADIR	Int	-1,1	-1 Branching down is preferred.
			1* Branching up is preferred.
BOUSTR	Int	0..1	0* Depth first strategy.
			1 Width first strategy.
BRASTR	Int	0..1	0 Use the average of lower and upper bound in father node as branching value.
			1* Use the weighted average of the scenario solutions of the component as branching value.
EPSILO	Dbl	0..	0 Specifies disjoint subdomains if continuous components are branched.
NULLDI	Dbl	-1..	0 Branch nodes only if their dispersion norm is greater than NULLDI.
ACCURA	Dbl	1e-18..1	1e-8 Accuracy – real values are considered equal if the absolute value of their difference is less than accuracy
HEURIS	Int	0..13,99	3 choice of heuristics, cf. Table 5

Table 3: Branch-and-bound parameters (1)

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
RELAXL	Int	0..2	0* No integrality relaxation.
			1 Relax first-stage variables.
			2 Relax first- and second-stage variables.
INTFIR	Int	0..1	0* Branching exclusively according to branching rules.
			1 Branching on integers first.

Table 4: Branch-and-bound parameters (2)

The MAXINHERIT parameter specifies a maximal level of passing on 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. It saves half of the solutions 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 additional 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.

4.4 Heuristic

The decomposition algorithm uses a heuristic to guess feasible first-stage solutions based on the solutions of the 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.

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 next 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	Apply heuristic 3 and 5 alternating
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
12	Try solutions of all scenarios
13	Solve a randomly selected scenario with <i>high</i> precision, the remaining ones with <i>low</i>
99	Indicates that a list of heuristics is given in the sequel. See the example in the Appendix.

Table 5: Values of parameter HEURIS

4.5 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..	1	Weight on risk term in objective.
TARGET	Dbl	..	0	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 (standard deviation)).
			2	NFSD-algorithm.
BRAETA	Int	0..1	0	Branch order of additional first-stage variable in models 4 and 5. A value of 0 means that this variable will not be branched.
RISKBM	Dbl	..	1e+10	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 $\min_X EX + \alpha \mathbb{R}X$, negative ones the risk model $\min_X \mathbb{R}X$. These pure risk models are often harder to solve.

4.6 Parameters for the dual method

There is no description of `ConicBundle` available, yet. We refer to Helmberg (2000) where the used 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.

Name	Type	Range	Default/Description
CBFREQ	Int	0..	0 Use <code>ConicBundle</code> in every i th node.
CBITLI	Int	0..	1 Iteration limit: descent steps.
CBTOTI	Int	0..	1 Iteration limit: total iterations, i.e. descent and null steps.
CBPREC	Dbl	0..	1e-1 Precision of bundle method.
CBPRIN	Int	0..	0 <code>ConicBundle</code> 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..	50 Bundle size
CBMAXS	Int	1..	1 Maximal number of new subgradients
CBNEXT	Dbl	0..	Next weight

Table 8: `ConicBundle` parameters

5 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.

The first-stage variables have to be identified by a prefix or a postfix appended to their names as specified in the specs file.

Variables with stochastic cost coefficients have to be placed at the beginning and according to the order in which the stochastic cost coefficients are provided in the scenario file.

In the presence of stochastic right-hand sides, the constraints require a certain order, too. First-stage constraint have to be followed by the constraints with stochastic right-hand sides. Second-stage constraints without stochastic right-hand sides have to be placed at the end. The placing of first-stage constraints in the beginning is mandatory for the construction of the deterministic equivalent, too.

6 The scenario files

Unfortunately, the program `ddsip` is not conform to the SMPS data format, yet. The proprietary data format of the scenario files is described below.

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
scenario1	scenario1	position
0.1	23	200
23	34	186
34		
⋮	⋮	⋮
scenarioN	scenarioN	scenario1
0.1	30	23
30	41	34
41		
⋮	⋮	⋮
		scenarioN
		30
		41

Figure 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 contains only the probabilities of the individual scenarios.

In order to assign the stochastic right-hand sides to constraints, the constraints have to be grouped in the model file, cf. Section 3. The constraint section begins with the first-stage constraints followed by the stochastic constraints ordered corresponding to the entries in the scenario file, and closes with the second-stage constraints without stochastic right-hand sides.

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. The first c variables occurring in the objective function of the model file are assumed to have stochastic cost coefficients.

The matrix scenario file requires additionally a row and column index for each stochastic matrix coefficient, see Figure 3. The identifier for this index is `pos`. If m is the number of stochastic matrix entries, the $(2 * i - 1)$ -th number ($1 \leq i \leq m$) following `pos` is treated as row index and the $(2 * i)$ -th number as column index of the i -th entry of the individual scenarios.

Indices start with 0 for the first column/row. To facilitate the identification of the indices, a simple program `siphelp` is included in the sources, which reads a model file and produces a gzipped text file with a list of indices and names of rows and columns, named `rows+cols.gz`, and optionally an lp-file of the model.

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	
:	

Figure 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 **indices** (starting with 0 as in the output of `siphelp`) of first-stage variables and the second one integer values. High values lead to early branching.

8 Output

8.1 Output on screen

Figure 5 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 straight forward. 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.	Gap	Time
0	1 1	-85.4728	infeasible		-85.4728	2		0.01
1	3 2	-75.44188	infeasible		-85.4728	2		0.01
2	3 1	infeasible	infeasible		-75.4418	2		0.01
* 3	5 2	Heuristic 3	-56.6828	-56.6828				0.02
* 3	5 2	Heuristic 7	-63.636	-63.6365				0.02
3	5 2	-67.761944	-63.636	-63.6365	-75.4418	2	18.5%	0.02
4	5 2	-75.043288	infeasible	-63.6365	-75.0432	2	17.9%	0.02

Figure 5: Output lines

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 violating the nonanticipativity.
Gap is the gap between *Best Value* and *Bound*.
Time is the CPU time passed since invoking the program.

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 nonanticipativity 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 *sip.out*. 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 than 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`.

- The files *rhs.out*, *cost.out*, *matrix.out*, and *model.mps* offer a check for a correct reading of the scenario files and the model file, respectively.
- The files *risk.mps* and *eev.mps* document the changes made during defining the risk model and solving the expected value problem, respectively.
- The files *solution.out* contains information on the solution with the optimal first-stage solution among them.
- 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.

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 gollmer@math.uni-duisburg.de. If possible, provide the input files to facilitate reproduction of the error.

A Sample specs file

* Parameters to specify the two-stage model

FIRSTCON	9	* First stage constraints
FIRSTVAR	30	* First stage variables
SECCON	280	* Second stage constraints
SECVAR	326	* Second stage variables
STOCRHS	70	* Number of stochastic rhs elements
SCENARIOS	5	* Number of scenarios
STOCCOST	0	* Number of stochastic cost coefficients (stochastic objective doesn't work in combination with risk!)
STOCMAT	0	* Number of stochastic matrix entries

* Parameters for dual decomposition procedure

OUTLEVEL	5	* Print info to <i>more.out</i>
OUTFILES	1	* Print files (models and output)
STARTINFO	0	* Use start solution/bound
HOTSTART	1	* Use solution of previous subproblems doesn't work together with Conic Bundle!
PREPRO	0	* Sort scenarios
NODELIM	90	* <i>ddsip</i> node limit
TIMELIM	800	* <i>ddsip</i> time limit
HEURISTIC	4	* Heuristic 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
PORDER	0	* Use branching priority order
BRADIR	-1	* Branching direction
BRASTRAT	0	* Branching strategy
BOUSTRAT	1	* Bounding strategy
EPSILON	1e-4	* Epsilon used for branching on real variables
RELAXL	1	* Relaxation level (1= relax first-stage variables in subproblems)
LOGFREQ	1	* Output log frequency
ACCURACY	1e-8	* Accuracy used to compare real values
NULLDISP	0	* Tolerance level for null-dispersion (used together with ACCURACY!)
QUANTILES	10	Number of Quantiles to be output at the end

```

* 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 (models 4 and 5)?
* CPLEX Parameters (See CPLEX manual CPLEX (2002))

CPLEXBEGIN
1035          0 * Output on screen indicator
2009          0.31 * Relative gap
CPLEXEEV
2009          0.1 * Relative gap for EEV
CPLEXUB
1035          1 * Output on screen indicator for upper bounds
CPLEXDUAL
1035          1 * 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 Conic Bundle
CBTOTIT     1000 * Limit for the total number of Conic Bundle iterations (incl. nullsteps)
NONANT      2 * Nonanticipativity representation
CBPRINT     1 * Output level of Conic Bundle
CBBUNS      200 * Maximal bundle size
CBMAXS      10 * Maximal number of subgradients
CBNEXT      0 * Conic Bundle next weight

```

Specification file

References

- 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 (2004), <http://www-user.tu-chemnitz.de/~helmberg/>.
- C. Helmberg (2000), Semidefinite Programming for Combinatorial Optimization, ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin.
- ILOG (2002), ILOG CPLEX 8.0 Reference Manual, ILOG, Gentilly, France.
- 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. Ruszczyński, 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. Ruszczyński (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.