Math. Prog. Comp.
DOI 10.1007/s12532-012-0038-z

FULL LENGTH PAPER

A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition

Victor Zverovich · Csaba I. Fábián · Eldon F. D. Ellison · Gautam Mitra

Received: 21 December 2010 / Accepted: 12 March 2012

Abstract We report a computational study of two-stage SP models on a large set of benchmark problems and consider the following methods: (i) Solution of the deterministic equivalent problem by the simplex method and an interior point method, (ii) Benders decomposition (L-shaped method with aggregated cuts), (iii) Regularised decomposition of Ruszczyński (Math Program 35:309–333, 1986), (iv) Benders decomposition with regularization of the expected recourse by the level method (Lemaréchal et al. in Math Program 69:111–147, 1995), (v) Trust region (regularisation) method of Linderoth and Wright (Comput Optim Appl 24:207–250, 2003). In this study the three regularisation methods have been introduced within the computational structure of Benders decomposition. Thus second-stage infeasibility is controlled in the traditional manner, by imposing feasibility cuts. This approach allows extensions of the regularisation to feasibility issues, as in Fábián and Szőke (Comput Manag Sci 4:313–353, 2007). We report computational results for a wide range of benchmark

V. Zverovich (✉) · E. F. D. Ellison · G. Mitra
OptiRisk Systems, OptiRisk R&D House, One Oxford Road, Uxbridge, Middlesex, UB9 4DA, UK
e-mail: victor@optirisk-systems.com

E. F. D. Ellison · G. Mitra
The Centre for the Analysis of Risk and Optimisation Modelling Applications (CARISMA), School of Information Systems, Computing and Mathematics, Brunel University, London, UK
e-mail: masrefe@brunel.ac.uk

C. I. Fábián
Institute of Informatics, Kecskemét College, 10 Izsáki út, Kecskemét 6000, Hungary
e-mail: fabian.csaba@gamf.kefo.hu

C. I. Fábián
Department of OR, Loránd Eötvös University, Budapest, Hungary

problems from the POSTS and SLPTESTSET collections and a collection of difficult test problems compiled by us. Finally the scale-up properties and the performance profiles of the methods are presented.

Mathematics Subject Classification 49M27 · 65K05 · 90C05 · 90C06 · 90C15 · 90C51

1 Introduction and background

Formulation of stochastic optimisation problems and computational algorithms for their solution continue to make steady progress as can be seen from an analysis of many developments in this field. The edited volume by Wallace and Ziemba [45] outlines both the modelling systems for stochastic programming (SP) and many applications in diverse domains.

More recently, Fabozzi et al. [18] have considered the application of SP models to challenging financial engineering problems. The tightly knit yet highly focused Stochastic Programming Community, their active website http://stoprog.org, and their triennial international SP conference points to the progressive acceptance of SP as a valuable decision tool. The Committee on Stochastic Programming (COSP) exists as a standing committee of the Mathematical Optimization Society, and also serves as a liaison to related professional societies to promote stochastic programming.

At the same time many of the major software vendors, namely, XPRESS, AIMMS, MAXIMAL, and GAMS have started offering SP extensions to their optimisation suites.

Our analysis of the modelling and algorithmic solver requirements reveals that (a) modelling support, (b) scenario generation and (c) solution methods are three important aspects of a working SP system. Our research is focused on all three aspects and we refer the readers to Valente et al. [42] for modelling and Mitra et al. [32] and Di Domenica et al. [13] for scenario generation. In this paper we are concerned entirely with computational solution methods. Given the tremendous advances in LP solver algorithms there is a certain amount of complacency that by constructing a "deterministic equivalent" problem it is possible to process most realistic instances of SP problems. In this paper we highlight the shortcoming of this line of argument. We describe the implementation and refinement of established algorithmic methods and report a computational study which clearly underpins the superior scale up properties of the solution methods which are described in this paper.

A taxonomy of the important class of SP problems may be found in [42,43]. The most important class of problems with many applications is the two-stage stochastic programming model with recourse; this class of models originated from the early research of Beale [2], Dantzig [10] and Wets [46].

A comprehensive treatment of the models and solution methods can be found in [7,25,31,35,40], and [24]. Some of these monographs contain extensions of the original model. Colombo et al. [9] and Gassmann and Wallace [20] describe computational studies which are based on interior point method and simplex based methods respectively. Birge [4] covers a broad range of SP solution algorithms and applications in his survey.

The rest of this paper is organised in the following way. In Sect. 2 we introduce the model setting of the two stage stochastic programming problem, in Sect. 3 we consider a selection of solution methods for processing this class of problems. First we consider direct solution of the deterministic equivalent LP problem. Then we discuss Benders decomposition, and the need for regularisation. We first present the Regularized Decomposition method of Ruszczyński [37] and then introduce another regularisation approach in some detail, namely, the Level Method [29] adapted for the two-stage stochastic programming problem. Finally we outline the box-constrained trust-region method of Linderoth and Wright [30].

In Sect. 4 we discuss implementation issues, in Sect. 5 we set out the computational study and in Sect. 6 we summarise our conclusions.

2 The model setting

2.1 Two-stage problems

In this paper we only consider linear SP models and assume that the random parameters have a discrete finite distribution. This class is based on two key concepts of (i) a finite set of discrete scenarios (of model parameters) and (ii) a partition of variables to first stage ("here and now") decision variables and a second stage observation of the parameter realisations and corrective actions and the corresponding recourse (decision) variables.

The first stage decisions are represented by the vector $\boldsymbol{x}$. Assume there are S possible outcomes (*scenarios*) of the random event, the ith outcome occurring with probability p_i. Suppose the first stage decision has been made with the result $\boldsymbol{x}$, and the ith scenario is realised. The second stage decision $\boldsymbol{y}$ is computed by solving the following *second-stage problem* or *recourse problem*

$$\begin{aligned} R_i(\boldsymbol{x}): \quad & \min \quad \boldsymbol{q_i^T y} \\ & \text{subject to} \quad T_i \boldsymbol{x} + W_i \boldsymbol{y} = \boldsymbol{h_i}, \\ & \qquad\qquad \boldsymbol{y} \geq \boldsymbol{0}, \end{aligned} \tag{1}$$

where $\boldsymbol{q_i}$, $\boldsymbol{h_i}$ are given vectors and T_i, W_i are given matrices. Let K_i denote the set of those $\boldsymbol{x}$ vectors for which the recourse problem $R_i(\boldsymbol{x})$ has a feasible solution. This is a convex polyhedron. For $\boldsymbol{x} \in K_i$, let $q_i(\boldsymbol{x})$ denote the optimal objective value of the recourse problem. We assume that $q_i(\boldsymbol{x}) > -\infty$. (Or equivalently, we assume that the dual of the recourse problem $R_i(\boldsymbol{x})$ has a feasible solution. Solvability of the dual problem does not depend on $\boldsymbol{x}$.) The function $q_i : K_i \to \mathbb{R}$ is a *polyhedral* (i.e., piecewise linear) convex function.

The customary formulation of the *first-stage problem* is stated as:

$$\begin{aligned} \min \quad & \boldsymbol{c^T x} + \sum_{i=1}^{S} p_i \, q_i(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in X, \\ & \boldsymbol{x} \in K_i \quad (i = 1, \ldots, S), \end{aligned} \tag{2}$$

where $X := \{\boldsymbol{x} \,|\, A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}\}$ is a non-empty bounded polyhedron describing the constraints, $\boldsymbol{c}$ and $\boldsymbol{b}$ are given vectors and A is a given matrix, with compatible sizes. The expectation part of the objective, $F(\boldsymbol{x}) := \sum_{i=1}^{S} p_i\, q_i(\boldsymbol{x})$, is called the *expected recourse function*. This is a polyhedral convex function with the domain $K := K_1 \cap \cdots \cap K_S$.

This two-stage stochastic programming problem ((2) and (1)) can be formulated as a single linear programming problem called the *deterministic equivalent problem*:

$$\begin{array}{rl} \min & \boldsymbol{c^T x} + p_1 \boldsymbol{q_1^T y_1} + \ldots + p_S \boldsymbol{q_S^T y_S} \\ \text{subject to} & A\boldsymbol{x} \qquad\qquad\qquad\qquad\quad = \boldsymbol{b}, \\ & T_1\boldsymbol{x} + \ W_1 \boldsymbol{y_1} \qquad\qquad\quad = \boldsymbol{h_1}, \\ & \vdots \qquad\qquad\quad \ddots \qquad\quad \vdots \\ & T_S\boldsymbol{x} + \qquad\qquad\quad W_S \boldsymbol{y_S} \ = \boldsymbol{h_S}, \\ & \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{y_1} \geq \boldsymbol{0},\ \ldots,\ \boldsymbol{y_S} \geq \boldsymbol{0}. \end{array} \tag{3}$$

3 A selection of methods

3.1 Solution of the deterministic equivalent problem

The deterministic equivalent problem (3) can be solved as a linear programming problem, either by the simplex method or by an interior-point method. In this computational study we use general-purpose LP solvers for the solution of the deterministic equivalent problem.

3.2 Decomposition methods

The deterministic equivalent problem (3) is a linear programming problem of a specific structure: for each scenario, a subproblem is included that describes the second-stage decision associated with the corresponding scenario realisation. The subproblems are linked by the first-stage decision variables. Dantzig and Madansky [11] observed that the dual of the deterministic equivalent problem fits the prototype for the Dantzig–Wolfe decomposition [12].

Van Slyke and Wets [44] proposed a cutting-plane approach for the first-stage problem (2). Their L-Shaped method builds respective cutting-plane models of the feasible domain $K = K_1 \cap \cdots \cap K_S$ and of the expected recourse function $F = \sum_{i=1}^{S} p_i q_i$. We outline cutting-plane models and their relationship with decomposition.

Let us denote the dual of $R_i(\boldsymbol{x})$ in (1) as

$$\begin{array}{rll} D_i(\boldsymbol{x}): & \max & \boldsymbol{z^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \\ & \text{subject to} & W_i^T \boldsymbol{z} \leq \boldsymbol{q_i}, \end{array} \tag{4}$$

where $\boldsymbol{z}$ is a real-valued vector.

The feasible region is a convex polyhedron that we assumed nonempty. We will characterise this polyhedron by two finite sets of vectors: let U_i and V_i denote the

sets of the extremal points and of the extremal rays, respectively, in case the polyhedron can be represented by these sets. To handle the general case, we require further formalism; let us add a slack vector $\boldsymbol{\gamma}$ of appropriate dimension, and use the notation $[W_i^T, I](z, \boldsymbol{\gamma}) = W_i^T z + \boldsymbol{\gamma}$. Given a composite vector $(z, \boldsymbol{\gamma})$ of appropriate dimensions, let $\mathrm{support}(z, \boldsymbol{\gamma})$ denote the set of those column-vectors of the composite matrix $[W_i^T, I]$ that belong to non-zero $(z, \boldsymbol{\gamma})$-components. Using these, let

$$U_i := \left\{ z \,\middle|\, W_i^T z + \boldsymbol{\gamma} = \boldsymbol{q_i}, \boldsymbol{\gamma} \geq \mathbf{0}, \ \mathrm{support}(z, \boldsymbol{\gamma}) \text{ is a linearly independent set} \right\},$$
$$V_i := \left\{ z \,\middle|\, W_i^T z + \boldsymbol{\gamma} = \mathbf{0}, \boldsymbol{\gamma} \geq \mathbf{0}, \|(z, \boldsymbol{\gamma})\| = 1, \ \mathrm{support}(z, \boldsymbol{\gamma}) \text{ is a minimal dependent set} \right\},$$

where a minimal dependent set is a set that is not linearly independent; but is minimal in the sense that having discarded any of its elements, the remaining elements compose a linearly independent set.

U_i is the set of the basic feasible solutions of problem (4) and hence it is a finite set. Finiteness of V_i can be proven in a similar manner: Given a minimal dependent subset of the columns of the matrix $[W_i^T, I]$, there are no more than 2 vectors in V_i that have the given subset as support. The feasible domain of the dual problem $D_i(\boldsymbol{x})$ in (4) can be represented as convex combinations of U_i-elements added to cone-combinations of V_i-elements.

We have $\boldsymbol{x} \in K_i$ if and only if the dual problem $D_i(\boldsymbol{x})$ has a finite optimum, that is,

$$\boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq 0 \quad \text{holds for every } \boldsymbol{v_i} \in V_i.$$

In this case, the optimum of $D_i(\boldsymbol{x})$ is attained at an extremal point, and can be computed as

$$\begin{aligned} \min \quad & \vartheta_i \\ \text{subject to} \quad & \vartheta_i \in \mathbb{R}, \\ & \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq \vartheta_i \quad (\boldsymbol{u_i} \in U_i). \end{aligned}$$

By the linear programming duality theorem, the optimum of the above problem is equal to $q_i(\boldsymbol{x})$; hence the first-stage problem (2) is written as

$$\begin{aligned} \min \quad & \boldsymbol{c^T x} + \sum_{i=1}^{S} p_i \vartheta_i \\ \text{subject to} \quad & \boldsymbol{x} \in X, \quad \vartheta_i \in \mathbb{R} \qquad (i = 1, \ldots, S), \\ & \boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq 0 \quad (\boldsymbol{v_i} \in V_i, \ i = 1, \ldots, S), \\ & \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq \vartheta_i \ (\boldsymbol{u_i} \in U_i, \ i = 1, \ldots, S); \end{aligned} \tag{5}$$

This we call the *disaggregated form*. The *aggregated form* is stated as

$$\begin{aligned} \min \quad & \boldsymbol{c^T x} + \vartheta \\ \text{subject to} \quad & \boldsymbol{x} \in X, \quad \vartheta \in \mathbb{R}, \\ & \boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq 0 \qquad (\boldsymbol{v_i} \in V_i, \ i = 1, \ldots, S), \\ & \sum_{i=1}^{S} p_i \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq \vartheta \ \left((\boldsymbol{u_1}, \ldots, \boldsymbol{u_S}) \in \mathcal{U} \right), \end{aligned} \tag{6}$$

where $\mathcal{U} \subset U_1 \times \cdots \times U_S$ is a subset that contains an element for each facet in the graph of the polyhedral convex function F; formally, we have

$$F(\boldsymbol{x}) = \sum_{i=1}^{S} \left\{ p_i \max_{\boldsymbol{u_i} \in U_i} \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}) \right\} = \max_{(\boldsymbol{u_1},\ldots,\boldsymbol{u_S}) \in \mathcal{U}} \sum_{i=1}^{S} p_i \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i\boldsymbol{x}).$$

Cutting-plane methods can be devised on the basis of either the disaggregated formulation (5) or the aggregated formulation (6). These are processed by iterative methods that build respective cutting-plane models of the feasible set K and the expected recourse function F. Cuts at a given iterate $\widehat{\boldsymbol{x}}$ are generated by solving the dual problems $D_i(\widehat{\boldsymbol{x}})$ $(i = 1, \ldots, S)$. Dual problems with unbounded objectives yield *feasibility cuts* that are used to construct a model of K. Dual problems with optimal solutions yield *optimality cuts* that are used to construct a model of F.

In its original form, the L-Shaped method of Van Slyke and Wets [44] works on the aggregated problem. A *multicut* version that works on the disaggregated problem was proposed by Birge and Louveaux [6].

There is a close relationship between decomposition and cutting-plane approaches. It turns out that the following approaches yield methods that are in principle identical:

- cutting-plane method for either the disaggregated problem (5) or the aggregated problem (6),
- Dantzig–Wolfe decomposition [12] applied to the dual of the deterministic equivalent problem (3),
- Benders decomposition [3] applied to the deterministic equivalent problem (3).

Cutting-plane formulations have the advantage that they give a clear visual illustration of the procedure. A state-of-the-art overview of decomposition methods can be found in [38].

Aggregated versus disaggregated formulations

The difference between the aggregated and the disaggregated problem formulations may result in a substantial difference in the efficiency of the solution methods. By using disaggregated cuts, more detailed information is stored in the master problem, hence the number of the master iterations is reduced in general. This is done at the expense of larger master problems.

Birge and Louveaux [7] conclude that the multicut approach is in general more effective when the number of the scenarios is not significantly larger than the number of the constraints in the first-stage problem. This conclusion is based on their own numerical results [6] and those of Gassmann [19].

3.3 Regularisation and trust region methods

It is observed that successive iterations do not generally produce an orderly progression of solutions—in the sense that while the change in objective value from one iteration

to the next may be very small, even zero, a wide difference may exist between corresponding values of the first-stage variables. This feature of zigzagging in cutting plane methods is the consequence of using a linear approximation. Improved methods were developed that use quadratic approximation: proximal point method by [36], and bundle methods by [26] and [28]. These methods construct a sequence of *stability centers* together with the sequence of the iterates. When computing the next iterate, roaming away from the current stability center is penalised.

Another approach is the trust region methods, where a trust region is constructed around the current stability center, and the next iterate is selected from this trust region.

Regularized decomposition

The Regularized Decomposition (RD) method of Ruszczyński [37] is a bundle-type method applied to the minimisation of the sum of polyhedral convex functions over a convex polyhedron, hence this method fits the disaggregated problem (5). The RD method lays an emphasis on keeping the master problem as small as possible. (This is achieved by an effective constraint reduction strategy.) A recent discussion of the RD method can be found in [38].

Ruszczyński and Świętanowski [41] implemented the RD method, and solved two-stage stochastic programming problems, with a growing scenario set. Their test results show that the RD method is capable of handling large problems.

The level method

A more recent development in convex programming is the level method of Lemaréchal et al. [29]. This is a special bundle-type method that uses level sets of the model functions for regularisation. Let us consider the problem

$$\begin{array}{c} \min \ f(\boldsymbol{x}) \\ \text{subject to} \\ \boldsymbol{x} \in Y, \end{array} \tag{7}$$

where $Y \subset \mathbb{R}^n$ is a convex bounded polyhedron, and f a real-valued convex function, Lipschitzian relative to Y. The level method is an iterative method, a direct generalization of the classical cutting-plane method. A *cutting-plane model* of f is maintained using function values and subgradients computed at the known iterates. Let $\underline{f}$ denote the current model function; this is the upper cover of the linear support functions drawn at the known iterates. Hence $\underline{f}$ is a polyhedral convex lower approximation of f. The level sets of the model function are used for regularization.

Let $\hat{\boldsymbol{x}}$ denote the current iterate. Let $F^\star$ denote the minimum of the objective values in the known iterates. Obviously $F^\star$ is an upper bound for the optimum of (7).

Let $\underline{F} := \min_{\boldsymbol{x} \in Y} \underline{f}(\boldsymbol{x})$ denote the minimum of the current model function over the feasible polyhedron. Obviously $\underline{F}$ is a lower bound for the optimum of (7).

If the gap $F^\star - \underline{F}$ is small, then the algorithm stops. Otherwise let us consider the level set of the current model function belonging to the level $(1-\lambda)F^\star + \lambda\underline{F}$ where $0 < \lambda < 1$ is a fixed parameter. Using formulas, the current level set is

$$\hat{Y} := \left\{ \boldsymbol{x} \in Y \;\middle|\; \underline{f}(\boldsymbol{x}) \leq (1-\lambda)F^{\star} + \lambda \underline{F} \right\}.$$

The next iterate is obtained by *projecting* the current iterate onto the current level set. Formally, the next iterate is an optimal solution of the convex quadratic programming problem $\min \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$ subject to $\boldsymbol{x} \in \hat{Y}$.

Lemaréchal et al. [29] gives the following efficiency estimate: To obtain a gap smaller than ϵ, it suffices to perform

$$\kappa \left(\frac{DL}{\epsilon} \right)^2 \tag{8}$$

iterations, where D is the diameter of the feasible polyhedron, L is a Lipschitz constant of the objective function, and κ is a constant that depends only on the parameter of the algorithm.

Applying the level method to our first-stage problem (2), let $f(\boldsymbol{x}) := \boldsymbol{c}^T \boldsymbol{x} + F(\boldsymbol{x})$ and $Y := X \cap K$. The feasible domain is not known explicitly (except for problems with relatively complete recourse). Hence, in general, we must construct a cutting-plane model of Y using feasibility cuts. The level method must be adapted accordingly: the objective value can only be computed for feasible iterates. We clearly obtain a finite procedure because the set of the possible cuts is finite. (We never discard cuts in our implementation. Though there are means of bundle reduction for the level method, we did not implement them because the level method solved our test problems in but a few iterations.)

Remark 1 In case of relatively complete recourse no feasibility cuts are needed, and the efficiency estimate (8) applies. This estimate is essentially different from the classic finiteness results obtained when a polyhedral convex function is minimised by a cutting-plane method. Finiteness results are based on enumeration. The straightforward finiteness proof assumes that basic solutions are found for the model problems, and that there is no degeneracy. (These assumptions facilitate bundle reduction.)

An interesting finiteness proof that allows for nonbasic solutions is presented in [39]. This is based on the enumeration of the *cells* (i.e., polyhedrons, facets, edges, vertices) that the linear pieces of the objective function define.

Remark 2 In general, the constants L and D in (8) are not easily derived from the problem data. Upper bounds of the Lipschitz constant L are proposed in [17] for the case of special two-stage stochastic programming problems, e.g., those having network recourse (SP problems where the second-stage subproblems are network flow problems). But even with constants of reasonable magnitudes, the estimate (8) generally yields bounds too large for practical purposes.

However, the level method performs much better in practice than the estimate (8) implies. Nemirovski [33, chapter 5.3.2] observes the following experimental fact: when solving a problem with n variables, every n steps add a new accurate digit in our estimate of the optimum. This observation is confirmed by those experiments reported in [17], where the level method is applied for the solution of two-stage stochastic programming problems with relatively complete recourse. That paper also reports on solving the problems with growing scenario sets. According to the results presented,

there is no relationship connecting the number of the scenarios and the number of the level master iterations required (provided the number of the scenarios is large enough).

Remark 3 In the present study we have implemented the level method as described above. However, there are extensions of the level method that particularly fit in with a two-stage stochastic problem solver.

The constrained level method of Lemaréchal et al. [29] is a primal-dual method that solves convex problems involving constraint functions. The first-stage problem (2) can be formulated using a constraint function instead of the set constraints $\boldsymbol{x} \in K_i$ $(i = 1, \ldots, S)$.
A measure of the second-stage infeasibility can be used as a constraint function; namely, the expectation of the infeasibility in the corresponding second-stage problems. Fábián and Szőke [17] applied the constrained level method to this constraint-function formulation of the first-stage problem. The advantage of this approach is that regularisation extends to feasibility issues. (This approach requires extending the expected recourse function beyond K.)

Fábián [16] proposed inexact versions of the level method and the constrained level method. The inexact methods use approximate data to construct models of the objective and constraint functions. At the beginning of the procedure, a rough approximation is used, and the accuracy is gradually increased as the optimum is approached. Solving the first-stage problem with an inexact method facilitates a progressive approximation of the distribution. Moreover we can work with approximate solutions of the second-stage problems. Numerical results of Fábián and Szőke [17] show that this progressive approximation framework is effective: although the number of the master iterations is larger than in the case of the exact method, there is a substantial reduction in the solution time of the second-stage problems.

A different approach of using inexact bundle methods for two-stage stochastic programming problems is proposed by Oliveira et al. [34].

Remark 4 The level method can also be implemented for problems with unbounded domain Y. A set of initial cuts is then needed to make the master objective bounded from below. (Just like with a pure cutting-plane method.)

The constant D is never actually used in course of the level method; it is used only in the convergence proof and in the theoretical efficiency estimate (8). In case the objective function f is polyhedral, then finiteness of the procedure follows from the finiteness of the set of the possible cuts.

Remark 5 An interesting feature of the level method is that the parameter λ is fixed. (This is in contrast with other bundle-type methods that need continual tuning of the parameters in course of operation.)

We tested with different settings of λ, and experienced slight differences in running times, but could not find a universally best setting. We decided to work with $\lambda = 0.5$.

Box-constrained method

The box-constrained trust-region method of Linderoth and Wright [30] solves the disaggregated problem (5), and uses a special trust-region approach.

Trust-region methods construct a sequence of stability centers together with the sequence of the iterates. Trust regions are constructed around the stability centers, and the next iterate is selected from the current trust region. Linderoth and Wright construct box-shaped trust regions, hence the resulting master problems remain linear. The size of the trust region is continually adapted on the basis of the quality of the current solution.

4 Algorithmic descriptions and implementation issues

All solution methods considered in the current study were implemented within the FortSP stochastic solver system [15] which includes an extensible algorithmic framework for creating decomposition-based methods. The following algorithms were implemented based on this framework:

- Benders decomposition (Benders),
- Benders decomposition with regularisation by the level method (Level),
- the trust region method based on l_∞ norm (TR),
- regularized decomposition (RD).

For more details including the solver system architecture and pseudo-code of each method refer to [15]. Here we present only the most important details specific to our implementation.

4.1 Solution of the deterministic equivalent by simplex and interior-point methods

The first approach to solve stochastic linear programming problems we considered was using a state-of-the-art LP solver to optimise the deterministic equivalent problem (3). For this purpose CPLEX barrier and dual simplex optimisers were selected since they provide high-performance implementation of corresponding methods.

We also solved our problems by the HOPDM solver [8,21], an implementation of the infeasible primal-dual interior point method.

The results summarised in Table 1 show that while it took HOPDM on average less iterations to solve a problem, CPLEX barrier optimiser was faster in our benchmarks. This can be explained by the fact that CPLEX is optimised for the underlying hardware; in particular, it uses high performance Intel Math Kernel Library which is tuned for the hardware we were using in the tests.

4.2 Benders decomposition

For the present computational study, we have implemented a decomposition method that works on the aggregated problem (6). After a certain number of itera-

Table 1 Summary of CPLEX barrier optimiser and HOPDM performance on 95 test problems described in Sect. 5.2

	CPLEX	HOPDM
Average iterations	29	21
Average time	56.66	170.50
Solved problems	87	78

tions, let $\widehat{V}_i \subset V_i$ denote the subsets of the known elements of V_i $(i = 1, \ldots, S)$, respectively. Similarly, let $\widehat{\mathcal{U}} \subset \mathcal{U}$ denote the subset of the known elements of $\mathcal{U} \subset U_1 \times \cdots \times U_S$. We solve the current problem

$$\begin{array}{rl} \min & \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \vartheta \\ \text{subject to} & \boldsymbol{x} \in X, \quad \vartheta \in \mathbb{R}, \\ & \boldsymbol{v}_{\boldsymbol{i}}^{\boldsymbol{T}}(\boldsymbol{h}_{\boldsymbol{i}} - T_i\boldsymbol{x}) \le 0 \qquad (\boldsymbol{v}_{\boldsymbol{i}} \in \widehat{V}_i,\ i = 1, \ldots, S), \\ & \sum_{i=1}^{S} p_i \boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}(\boldsymbol{h}_{\boldsymbol{i}} - T_i\boldsymbol{x}) \le \vartheta \quad \left((\boldsymbol{u}_{\boldsymbol{1}}, \ldots, \boldsymbol{u}_{\boldsymbol{S}}) \in \widehat{\mathcal{U}}\right). \end{array} \tag{9}$$

If the problem (9) is infeasible, then so is the original problem. Let $\widehat{\boldsymbol{x}}$ denote an optimal solution. In order to generate cuts at $\widehat{\boldsymbol{x}}$, we solve the dual recourse problems $D_i(\widehat{\boldsymbol{x}})$ $(i = 1, \ldots, S)$ with a simplex-type method. Let

$$\widehat{I} := \left\{1 \le i \le S \,\middle|\, \text{problem } D_i(\widehat{\boldsymbol{x}}) \text{ has unbounded objective}\right\}.$$

If $\widehat{I} = \emptyset$ then the solution process of each dual recourse problem terminated with an optimal basic solution $\widehat{\boldsymbol{u}}_{\boldsymbol{i}} \in U_i$. If $\widehat{\boldsymbol{x}}$ is near-optimal then the procedure stops. Otherwise we add the point $(\widehat{\boldsymbol{u}}_{\boldsymbol{1}}, \ldots, \widehat{\boldsymbol{u}}_{\boldsymbol{S}})$ to $\widehat{\mathcal{U}}$, rebuild the model problem (9), and start a new iteration.

If $\widehat{I} \neq \emptyset$ then for $i \in \widehat{I}$, the solution process of the dual recourse problem $D_i(\widehat{\boldsymbol{x}})$ terminated with $\widehat{\boldsymbol{v}}_{\boldsymbol{i}} \in V_i$. We add $\widehat{\boldsymbol{v}}_{\boldsymbol{i}}$ to $\widehat{V}_i$ $(i \in \widehat{I})$, rebuild the problem (9), and start a new iteration.

4.3 Benders decomposition with level regularisation

On the basis of the decomposition method described above we implemented a rudimentary version of the level decomposition. We use the original exact level method, hence we use no distribution approximation, and second-stage problems are solved exactly (i.e., with the same high accuracy always). Algorithm 1 shows the pseudo-code for the method.

Our computational results reported in Sect. 5.3 show that level-type regularisation is indeed advantageous.

4.4 Regularized decomposition

In addition to the methods that work on the aggregated problem we implemented two algorithms based on the disaggregated (multicut) formulation (5).

The first method is Regularized Decomposition (RD) of Ruszczyński [37]. For this method we implemented deletion of inactive cuts and the rules for dynamic adaptation of the penalty parameter σ as described by Ruszczyński and Świętanowski [41]:

- if $F(\boldsymbol{x}^k) > \gamma F(\bar{\boldsymbol{x}}^k) + (1-\gamma)\hat{F}^k$ then $\sigma \leftarrow \sigma/2$,
- if $F(\boldsymbol{x}^k) < (1-\gamma)F(\bar{\boldsymbol{x}}^k) + \gamma\hat{F}^k$ then $\sigma \leftarrow 2\sigma$,

Algorithm 1 Benders decomposition with regularisation by the level method

choose iteration limit $k_{max} \in \mathbb{Z}_+$
choose relative stopping tolerance $\epsilon \in \mathbb{R}_+$
solve the expected value problem to get a solution $\boldsymbol{x}^0$ (initial iterate)
$k \leftarrow 0$, $F^* \leftarrow \infty$
choose $\lambda \in (0, 1)$
$F^0 \leftarrow -\infty$
while time limit is not reached **and** $k < k_{max}$ **do**
  solve the recourse problems (1) with $\boldsymbol{x} = \boldsymbol{x}^k$ and compute $F(\boldsymbol{x}^k)$
  if all recourse problems are feasible **then**
    add an optimality cut
    if $F(\boldsymbol{x}^k) < F^*$ **then**
      $F^* \leftarrow F(\boldsymbol{x}^k)$
      $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^k$
    end if
  else
    add a feasibility cut
  end if
  if $(F^* - F^k)/(|F^*| + 10^{-10}) \le \epsilon$ **then**
    stop
  end if
  solve the master problem (9) to get an optimal solution $(\boldsymbol{x}', \boldsymbol{\vartheta}')$ and the optimal objective value F^{k+1}.
  if $(F^* - F^{k+1})/(|F^*| + 10^{-10}) \le \epsilon$ **then**
    stop
  end if
  solve the projection problem:

$$\begin{aligned} \min\ & \|\boldsymbol{x} - \boldsymbol{x}'\|^2 \\ \text{subject to}\ & \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \vartheta \le (1-\lambda)F^{k+1} + \lambda F^* \\ & \boldsymbol{x} \in X, \quad \vartheta \in \mathbb{R}, \\ & \boldsymbol{v}_{\boldsymbol{i}}^{\boldsymbol{T}}(\boldsymbol{h}_{\boldsymbol{i}} - T_i\boldsymbol{x}) \le 0 \qquad (\boldsymbol{v}_{\boldsymbol{i}} \in \widehat{V}_i,\ i = 1, \ldots, S), \\ & \sum_{i=1}^{S} p_i \boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}(\boldsymbol{h}_{\boldsymbol{i}} - T_i\boldsymbol{x}) \le \vartheta \qquad \left((\boldsymbol{u}_{\boldsymbol{1}}, \ldots, \boldsymbol{u}_{\boldsymbol{S}}) \in \widehat{\mathcal{U}}\right). \end{aligned}$$

  let $(\boldsymbol{x}^{k+1}, \boldsymbol{\theta}^{k+1})$ be an optimal solution of the projection problem; then $\boldsymbol{x}^{k+1}$ is the next iterate
  $k \leftarrow k + 1$
end while

Here $\hat{F}^k = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x}^k + \vartheta^k$ and $F(\boldsymbol{x}) = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \sum_{i=1}^{S} p_i\, q_i(\boldsymbol{x})$.

where $\bar{\boldsymbol{x}}^k$ is a reference point, $(\boldsymbol{x}^k, \boldsymbol{\vartheta}^k)$ is a solution of the master problem at the iteration k, $\hat{F}^k = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x}^k + \sum_{i=1}^{S} p_i \vartheta_i^k$, $F(\boldsymbol{x}) = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \sum_{i=1}^{S} p_i\, q_i(\boldsymbol{x})$ and $\gamma \in (0, 1)$ is a parameter.

Regularised master problem at the iteration k is formulated as follows:

$$\begin{aligned} \min\ & \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \sum_{i=1}^{S} p_i \vartheta_i + \frac{1}{2\sigma}\|\boldsymbol{x} - \bar{\boldsymbol{x}}^{\boldsymbol{k}}\|^2 \\ \text{subject to}\ & \boldsymbol{x} \in X, \quad \vartheta_i \in \mathbb{R} \qquad (i = 1, \ldots, S), \\ & \boldsymbol{v}_{\boldsymbol{i}}^{\boldsymbol{T}}(\boldsymbol{h}_{\boldsymbol{i}} - T_i\boldsymbol{x}) \le 0 \qquad (\boldsymbol{v}_{\boldsymbol{i}} \in \widehat{V}_i,\ i = 1, \ldots, S), \\ & \boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}(\boldsymbol{h}_{\boldsymbol{i}} - T_i\boldsymbol{x}) \le \vartheta_i \qquad (\boldsymbol{u}_{\boldsymbol{i}} \in \widehat{U}_i,\ i = 1, \ldots, S). \end{aligned} \tag{10}$$

For a more detailed description of the implementation including the pseudo-code please refer to Ellison et al. [15].

4.5 The trust region method based on the infinity norm

We also implemented the l_∞ trust region L-shaped method of [30]. It operates on the disaggregated problem (5) with additional bounds of the form

$$-\Delta e \leq \boldsymbol{x} - \bar{\boldsymbol{x}}^{\boldsymbol{k}} \leq \Delta e, \tag{11}$$

where Δ is the trust region radius, $e = (1, 1, \ldots, 1)$ and $\bar{\boldsymbol{x}}^k$ is a reference point at the iteration k. The rules of updating Δ are the same as in [30] and are outlined below (counter is initially set to 0):

if $F(\bar{\boldsymbol{x}}^k) - F(\boldsymbol{x}^k) \geq \xi(F(\bar{\boldsymbol{x}}^k) - \hat{F}^k)$ **then**
  if $F(\bar{\boldsymbol{x}}^k) - F(\boldsymbol{x}^k) \geq 0.5(F(\bar{\boldsymbol{x}}^k) - \hat{F}^k)$ **and** $\|\bar{\boldsymbol{x}}^k - \boldsymbol{x}^k\|_\infty = \Delta$ **then**
    $\Delta \leftarrow \min(2\Delta, \Delta_{hi})$
  end if
  counter $\leftarrow 0$
else
  $\rho \leftarrow -\min(1, \Delta)(F(\bar{\boldsymbol{x}}^k) - F(\boldsymbol{x}^k))/(F(\bar{\boldsymbol{x}}^k) - \hat{F}^k)$
  if $\rho > 0$ **then**
    counter $\leftarrow$ counter + 1
  end if
  if $\rho > 3$ **or** (counter ≥ 3 **and** $\rho \in (1, 3]$) **then**
    $\Delta \leftarrow \Delta / \min(\rho, 4)$
    counter $\leftarrow 0$
  end if
end if

$\hat{F}^k = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x}^k + \sum_{i=1}^{S} p_i \vartheta_i^k$, where $(\boldsymbol{x}^k, \boldsymbol{\vartheta}^k)$ is a solution of the master problem at the iteration k, Δ_{hi} is an upper bound on the radius and $\xi \in (0, 1/2)$ is a parameter. The complete pseudo-code of the method can be found in Ellison et al. [15].

5 Computational study

5.1 Experimental setup

The computational experiments were performed on a Linux machine with 2.4 GHz Intel CORE i5 M520 CPU and 6 GiB of RAM. Deterministic equivalents were solved with CPLEX 12.1 dual simplex and barrier optimisers. Crossover to a basic solution was disabled for the barrier optimiser and the number of threads was limited to 1. For other CPLEX options the default values were used.

The times are reported in seconds with times of reading input files not included. For simplex and IPM the times of constructing deterministic equivalent problems are

included though it should be noted that they only amount to small fractions of the total. The time limit of 1,800 s was used in all the tests. CPLEX linear and quadratic programming solver was used to solve master problem and subproblems in the decomposition methods. All the test problems were presented in SMPS format introduced by Birge et al. [5].

The first-stage solution of the expected value problem was taken as a starting point for the decomposition methods. The values of the parameters are specified below.

- Benders decomposition with regularisation by the level method:
  $\lambda = 0.5$,
- Regularized decomposition:
  $\sigma = 1, \gamma = 0.9$.
- Trust region method based on l_∞ norm:
  $\Delta = 1, \Delta_{hi} = 10^3$ (except for the saphir problems where $\Delta_{hi} = 10^9$), $\xi = 10^{-4}$.

5.2 Data sets

We considered test problems which were drawn from four different sources described in Table 3. Table 2 gives the dimensions of these problems.

Most of the benchmark problems have stochasticity only in the right-hand side (RHS). Notable exception is the SAPHIR family of problems which has random elements both in the RHS and the constraint matrix. The first-stage subproblems are bounded in all our test problems.

It should be noted that the problems generated with GENSLP (rand0, rand1 and rand2) do not possess any internal structure inherent in real-world problems. However they are still useful for the purposes of comparing scale-up properties of algorithms.

5.3 Computational results

The computational results are presented in Tables 4 and 5. Iter denotes the number of iterations. For decomposition methods this is the number of master iterations.

Finally we present the results in the form of performance profiles. The performance profile for a solver is defined by Dolan and Moré [14] as the cumulative distribution function for a performance metric. We use the ratio of the solving time versus the best time as the performance metric. Let P and M be the set of problems and the set of solution methods respectively. We define by $t_{p,m}$ the time of solving problem $p \in P$ with method $m \in M$. For every pair (p, m) we compute performance ratio

$$r_{p,m} = \frac{t_{p,m}}{\min\{t_{p,m} | m \in M\}},$$

If method m failed to solve problem p the formula above is not defined. In this case we set $r_{p,m} := \infty$.

The cumulative distribution function for the performance ratio is defined as follows:

$$\rho_m(\tau) = \frac{|\{p \in P | r_{p,m} \leq \tau\}|}{|P|}$$

Table 2 Dimensions of test problems

Name	Scen	Stage 1		Stage 2		Deterministic equivalent		
		Rows	Cols	Rows	Cols	Rows	Cols	Nonzeros
fxm	6	92	114	238	343	1,520	2,172	12,139
	16	92	114	238	343	3,900	5,602	31,239
fxmev	1	92	114	238	343	330	457	2,589
pltexpa	6	62	188	104	272	686	1,820	3,703
	16	62	188	104	272	1,726	4,540	9,233
stormg2	8	185	121	528	1,259	4,409	10,193	27,424
	27	185	121	528	1,259	14,441	34,114	90,903
	125	185	121	528	1,259	66,185	157,496	418,321
	1,000	185	121	528	1,259	528,185	1,259,121	3,341,696
airl-first	25	2	4	6	8	152	204	604
airl-second	25	2	4	6	8	152	204	604
airl-randgen	676	2	4	6	8	4,058	5,412	16,228
assets	100	5	13	5	13	505	1,313	2,621
	37,500	5	13	5	13	187,505	487,513	975,021
4node	1	14	52	74	186	88	238	756
	2	14	52	74	186	162	424	1,224
	4	14	52	74	186	310	796	2,160
	8	14	52	74	186	606	1,540	4,032
	16	14	52	74	186	1,198	3,028	7,776
	32	14	52	74	186	2,382	6,004	15,264
	64	14	52	74	186	4,750	11,956	30,240
	128	14	52	74	186	9,486	23,860	60,192
	256	14	52	74	186	18,958	47,668	120,096
	512	14	52	74	186	37,902	95,284	239,904
	1,024	14	52	74	186	75,790	190,516	479,520
	2,048	14	52	74	186	151,566	380,980	958,752
	4,096	14	52	74	186	303,118	761,908	1,917,216
	8,192	14	52	74	186	606,222	1,523,764	3,834,144
	16,384	14	52	74	186	1,212,430	3,047,476	7,668,000
	32,768	14	52	74	186	2,424,846	6,094,900	15,335,712
4node-base	1	16	52	74	186	90	238	772
	2	16	52	74	186	164	424	1,240
	4	16	52	74	186	312	796	2,176
	8	16	52	74	186	608	1,540	4,048
	16	16	52	74	186	1,200	3,028	7,792
	32	16	52	74	186	2,384	6,004	15,280
	64	16	52	74	186	4,752	11,956	30,256
	128	16	52	74	186	9,488	23,860	60,208
	256	16	52	74	186	18,960	47,668	120,112

Table 2 continued

Name	Scen	Stage 1		Stage 2		Deterministic equivalent		
		Rows	Cols	Rows	Cols	Rows	Cols	Nonzeros
	512	16	52	74	186	37,904	95,284	239,920
	1,024	16	52	74	186	75,792	190,516	479,536
	2,048	16	52	74	186	151,568	380,980	958,768
	4,096	16	52	74	186	303,120	761,908	1,917,232
	8,192	16	52	74	186	606,224	1,523,764	3,834,160
	16,384	16	52	74	186	1,212,432	3,047,476	7,668,016
	32,768	16	52	74	186	2,424,848	6,094,900	15,335,728
4node-old	32	14	52	74	186	2,382	6,004	15,264
chem	2	38	39	46	41	130	121	289
chem-base	2	38	39	40	41	118	121	277
lands	3	2	4	7	12	23	40	92
lands-blocks	3	2	4	7	12	23	40	92
env-aggr	5	48	49	48	49	288	294	876
env-first	5	48	49	48	49	288	294	876
env-loose	5	48	49	48	49	288	294	876
env	15	48	49	48	49	768	784	2,356
	1,200	48	49	48	49	57,648	58,849	177,736
	1,875	48	49	48	49	90,048	91,924	277,636
	3,780	48	49	48	49	181,488	185,269	559,576
	5,292	48	49	48	49	254,064	259,357	783,352
	8,232	48	49	48	49	395,184	403,417	1,218,472
	32,928	48	49	48	49	1,580,592	1,613,521	4,873,480
env-diss-aggr	5	48	49	48	49	288	294	876
env-diss-first	5	48	49	48	49	288	294	876
env-diss-loose	5	48	49	48	49	288	294	876
env-diss	15	48	49	48	49	768	784	2,356
	1,200	48	49	48	49	57,648	58,849	177,736
	1,875	48	49	48	49	90,048	91,924	277,636
	3,780	48	49	48	49	181,488	185,269	559,576
	5,292	48	49	48	49	254,064	259,357	783,352
	8,232	48	49	48	49	395,184	403,417	1,218,472
	32,928	48	49	48	49	1,580,592	1,613,521	4,873,480
phone1	1	1	8	23	85	24	93	309
phone	32,768	1	8	23	85	753,665	2,785,288	9,863,176
stocfor1	1	15	15	102	96	117	111	447
stocfor2	64	15	15	102	96	6,543	6,159	26,907
rand0	2,000	50	100	25	50	50,050	100,100	754,501
	4,000	50	100	25	50	100,050	200,100	1,508,501

Table 2 continued

Name	Scen	Stage 1		Stage 2		Deterministic equivalent		
		Rows	Cols	Rows	Cols	Rows	Cols	Nonzeros
	6,000	50	100	25	50	150,050	300,100	2,262,501
	8,000	50	100	25	50	200,050	400,100	3,016,501
	10,000	50	100	25	50	250,050	500,100	3,770,501
rand1	2,000	100	200	50	100	100,100	200,200	3,006,001
	4,000	100	200	50	100	200,100	400,200	6,010,001
	6,000	100	200	50	100	300,100	600,200	9,014,001
	8,000	100	200	50	100	400,100	800,200	12,018,001
	10,000	100	200	50	100	500,100	1,000,200	15,022,001
rand2	2,000	150	300	75	150	150,150	300,300	6,758,501
	4,000	150	300	75	150	300,150	600,300	13,512,501
	6,000	150	300	75	150	450,150	900,300	20,266,501
	8,000	150	300	75	150	600,150	1,200,300	27,020,501
	10,000	150	300	75	150	750,150	1,500,300	33,774,501
saphir	50	32	53	8,678	3,924	433,932	196,253	1,136,753
	100	32	53	8,678	3,924	867,832	392,453	2,273,403
	200	32	53	8,678	3,924	1,735,632	784,853	4,546,703
	500	32	53	8,678	3,924	4,339,032	1,962,053	11,366,603
	1,000	32	53	8,678	3,924	8,678,032	3,924,053	22,733,103

Table 3 Sources of test problems

	Source	Reference	Comments
1.	POSTS collection	[22]	Two-stage problems from the (PO)rtable (S)tochastic programming (T)est (S)et (POSTS)
2.	Slptestset collection	[1]	Two-stage problems from the collection of stochastic LP test problems
3.	Random problems	[23]	Artificial test problems rand0, rand1 and rand2 generated with pseudo random stochastic LP problem generator GENSLP
4.	SAMPL problems	[27,43]	Problems instantiated from the SAPHIR gas portfolio planning model formulated in Stochastic AMPL (SAMPL)

We calculated performance profile of each considered method on the whole set of test problems. These profiles are shown in Fig. 1. The value of $\rho_m(\tau)$ gives the probability that method m solves a problem within a ratio τ of the best solver. For example according to Fig. 1 the level method was the first in 25 % of cases and solved 95 % of the problems within a ratio 11 of the best time.

The notable advantages of performance profiles over other approaches to performance comparison are as follows. Firstly, they minimize the influence of a small subset of problems on the benchmarking process. Secondly, there is no need to discard solver failures. Thirdly, performance profiles provide a visualisation of large sets of test

Table 4 Performance of DEP solution methods and level-regularised decomposition

Name	Scen	DEP-Simplex		DEP-IPM		Level		Optimal
		Time	Iter	Time	Iter	Time	Iter	Value
fxm	6	0.06	1,259	0.05	17	0.15	20	18,417.1
	16	0.22	3,461	0.13	23	0.15	20	18,416.8
fxmev	1	0.01	273	0.01	14	0.13	20	18,416.8
pltexpa	6	0.01	324	0.03	14	0.02	1	−9.47935
	16	0.01	801	0.08	16	0.02	1	−9.66331
stormg2	8	0.08	3,649	0.25	28	0.16	20	15,535,200
	27	0.47	12,770	2.27	27	0.31	17	15,509,000
	125	5.10	70,177	8.85	57	0.93	17	15,512,100
	1,000	226.70	753,739	137.94	114	6.21	21	15,802,600
airl-first	25	0.01	162	0.01	9	0.03	17	249,102
airl-second	25	0.00	145	0.01	11	0.03	17	269,665
airl-randgen	676	0.25	4,544	0.05	11	0.22	18	250,262
assets	100	0.02	494	0.02	17	0.03	1	−723.839
	37,500	1,046.85	190,774	6.37	24	87.55	2	−695.963
4node	1	0.01	110	0.01	12	0.06	21	413.388
	2	0.01	196	0.01	14	0.10	42	414.013
	4	0.01	326	0.02	17	0.11	45	416.513
	8	0.03	825	0.05	18	0.10	45	418.513
	16	0.06	1,548	0.11	17	0.15	44	423.013
	32	0.16	2,948	0.40	15	0.22	51	423.013
	64	0.72	7,185	0.44	17	0.36	54	423.013
	128	2.30	12,053	0.50	26	0.47	50	423.013
	256	7.69	31,745	1.05	30	0.87	48	425.375
	512	57.89	57,200	2.35	30	2.12	51	429.963
	1,024	293.19	133,318	5.28	32	3.95	53	434.112
	2,048	1,360.60	285,017	12.44	36	7.82	49	441.738
	4,096	t	–	32.67	46	9.12	46	446.856
	8,192	t	–	53.82	45	22.68	55	446.856
	16,384	t	–	113.20	46	45.24	52	446.856
	32,768	t	–	257.96	48	127.86	62	446.856
4node-base	1	0.01	111	0.01	11	0.04	16	413.388
	2	0.01	196	0.01	14	0.06	29	414.013
	4	0.01	421	0.02	14	0.07	30	414.388
	8	0.03	887	0.04	15	0.10	35	414.688
	16	0.06	1,672	0.11	17	0.10	30	414.688
	32	0.15	3,318	0.40	15	0.16	37	416.6
	64	0.49	7,745	0.36	13	0.22	33	416.6
	128	1.58	17,217	0.33	19	0.35	37	416.6

Table 4 continued

Name	Scen	DEP-Simplex		DEP-IPM		Level		Optimal
		Time	Iter	Time	Iter	Time	Iter	Value
	256	4.42	36,201	0.81	23	0.53	31	417.162
	512	22.44	80,941	2.20	29	1.45	37	420.293
	1,024	141.91	187,231	5.21	32	3.33	41	423.05
	2,048	694.89	337,082	11.12	32	6.13	42	423.763
	4,096	t	–	27.03	37	10.60	39	424.753
	8,192	t	–	51.29	40	24.99	48	424.775
	16,384	t	–	177.81	73	47.31	41	424.775
	32,768	t	–	242.91	48	102.29	49	424.775
4node-old	32	0.20	3,645	0.49	18	0.09	20	83,094.1
chem	2	0.00	29	0.00	11	0.03	15	−13,009.2
chem-base	2	0.00	31	0.00	11	0.05	14	−13,009.2
lands	3	0.00	21	0.00	9	0.02	10	381.853
lands-blocks	3	0.00	21	0.00	9	0.02	10	381.853
env-aggr	5	0.01	117	0.01	12	0.04	16	20,478.7
env-first	5	0.01	112	0.01	11	0.02	1	19,777.4
env-loose	5	0.01	112	0.01	12	0.02	1	19,777.4
env	15	0.01	321	0.01	16	0.05	15	22,265.3
	1,200	1.38	23,557	1.44	34	1.73	15	22,428.9
	1,875	2.90	36,567	2.60	34	2.80	15	22,447.1
	3,780	11.21	73,421	7.38	40	5.47	15	22,441
	5,292	20.28	102,757	12.19	42	7.67	15	22,438.4
	8,232	62.25	318,430	*m*	–	12.58	15	22,439.1
	32,928	934.38	1,294,480	*m*	–	75.67	15	22,439.1
env-diss-aggr	5	0.01	131	0.01	9	0.05	22	15,963.9
env-diss-first	5	0.01	122	0.01	9	0.04	12	14,794.6
env-diss-loose	5	0.01	122	0.01	9	0.03	5	14,794.6
env-diss	15	0.01	357	0.02	13	0.10	35	20,773.9
	1,200	1.96	26,158	1.99	50	2.80	35	20,808.6
	1,875	4.41	40,776	3.63	53	4.49	36	20,809.3
	3,780	16.94	82,363	9.32	57	8.87	36	20,794.7
	5,292	22.37	113,894	16.17	66	12.95	38	20,788.6
	8,232	70.90	318,192	*m*	–	22.49	41	20,799.4
	32,928	1,369.97	1,296,010	*m*	–	112.46	41	20,799.4
phone1	1	0.00	19	0.01	8	0.02	1	36.9
phone	32,768	t	–	50.91	26	48.23	1	36.9
stocfor1	1	0.00	39	0.01	11	0.03	6	−41,132
stocfor2	64	0.12	2,067	0.08	17	0.12	9	−39,772.4

Table 4 continued

Name	Scen	DEP-Simplex		DEP-IPM		Level		Optimal
		Time	Iter	Time	Iter	Time	Iter	Value
rand0	2,000	373.46	73,437	9.41	33	6.10	44	162.146
	4,000	1,603.25	119,712	34.28	62	10.06	32	199.032
	6,000	t	–	48.84	60	21.17	51	140.275
	8,000	t	–	56.89	49	28.86	50	170.318
	10,000	t	–	98.51	71	52.31	71	139.129
rand1	2,000	t	–	39.97	24	52.70	74	244.159
	4,000	t	–	92.71	28	72.30	59	259.346
	6,000	t	–	158.24	32	103.00	58	297.563
	8,000	t	–	228.68	34	141.81	65	262.451
	10,000	t	–	320.10	39	181.98	63	298.638
rand2	2,000	t	–	102.61	22	145.22	65	209.151
	4,000	t	–	225.71	24	170.08	42	218.247
	6,000	t	–	400.52	28	369.35	52	239.721
	8,000	t	–	546.98	29	369.01	44	239.158
	10,000	t	–	754.52	32	623.59	52	231.706
saphir	50	269.17	84,727	*n*	–	341.86	43	129,505,000
	100	685.50	152,866	*n*	–	700.44	46	129,058,000
	200	t	–	549.45	167	*t*	–	141,473,000
	500	t	–	*t*	–	608.48	44	137,871,000
	1,000	t	–	*n*	–	804.11	46	133,036,000

results as we have in our case. It should be noted, however, that we still investigated the failures and the cases of unusual performance. This resulted, in particular, in the adjustment of the values of ϵ, Δ_{hi} and ξ for the RD and TR methods and switching to a 64-bit platform with more RAM which was crucial for IPM.

As can be seen from Fig. 1, Benders decomposition with regularisation by the level method is both robust successfully solving the largest fraction of test problems and compares well with the other methods in terms of performance.

5.4 Comments on scale-up properties and on accuracy

We performed a set of experiments recording the change in the relative gap between the lower and upper bounds on objective function in the decomposition methods. The results are shown in Figs. 2, 3, 4 and 5. These diagrams show that level regularisation provides consistent reduction of the number of iterations needed to achieve the given precision. There are a few counterexamples, however, such as the env family of problems.

Figure 6 illustrates the scale-up properties of the algorithms in terms of the change in the solution time with the number of scenarios on the 4node problems. It shows that Benders decomposition with the level regularisation scales well at some point overtaking the multicut methods.

Table 5 Performance of decomposition methods

Name	Scen	Benders		Level		TR		RD	
		Time	Iter	Time	Iter	Time	Iter	Time	Iter
fxm	6	0.08	25	0.15	20	0.09	22	0.05	5
	16	0.09	25	0.15	20	0.11	22	0.07	5
fxmev	1	0.08	25	0.13	20	0.08	22	0.05	5
pltexpa	6	0.02	1	0.02	1	0.02	1	0.03	1
	16	0.02	1	0.02	1	0.02	1	0.03	1
stormg2	8	0.14	23	0.16	20	0.08	9	0.10	10
	27	0.47	32	0.31	17	0.18	10	0.23	11
	125	1.73	34	0.93	17	0.50	8	0.89	12
	1,000	11.56	41	6.21	21	3.38	6	7.30	11
airl-first	25	0.04	16	0.03	17	0.03	6	0.03	10
airl-second	25	0.02	10	0.03	17	0.02	4	0.03	5
airl-randgen	676	0.22	18	0.22	18	0.22	6	0.29	6
assets	100	0.02	1	0.03	1	0.03	1	0.02	1
	37,500	87.68	2	87.55	2	172.23	2	114.38	1
4node	1	0.03	24	0.06	21	0.03	8	0.03	15
	2	0.04	38	0.10	42	0.02	16	0.05	29
	4	0.04	41	0.11	45	0.03	14	0.05	19
	8	0.07	64	0.10	45	0.03	13	0.05	16
	16	0.11	67	0.15	44	0.04	12	0.05	13
	32	0.23	100	0.22	51	0.05	10	0.07	13
	64	0.27	80	0.36	54	0.08	11	0.12	14
	128	0.39	74	0.47	50	0.15	11	0.19	14
	256	0.95	71	0.87	48	0.20	7	0.29	9
	512	3.72	92	2.12	51	0.46	7	0.62	9
	1,024	5.14	70	3.95	53	0.42	3	1.23	10
	2,048	11.78	83	7.82	49	1.30	4	1.22	5
	4,096	18.46	89	9.12	46	2.79	3	2.03	4
	8,192	46.56	106	22.68	55	9.87	3	6.59	4
	16,384	99.00	110	45.24	52	38.28	3	27.50	4
	32,768	194.68	122	127.86	62	299.85	3	222.61	4
4node-base	1	0.03	31	0.04	16	0.03	21	0.03	14
	2	0.04	44	0.06	29	0.03	19	0.05	19
	4	0.06	58	0.07	30	0.04	20	0.07	34
	8	0.05	47	0.10	35	0.04	19	0.08	28
	16	0.08	56	0.10	30	0.06	21	0.11	28
	32	0.17	63	0.16	37	0.07	13	0.18	22
	64	0.23	61	0.22	33	0.17	19	0.30	21
	128	0.39	65	0.35	37	0.34	19	0.63	23
	256	0.89	66	0.53	31	0.45	11	1.81	26

Table 5 continued

Name	Scen	Benders		Level		TR		RD	
		Time	Iter	Time	Iter	Time	Iter	Time	Iter
	512	3.27	84	1.45	37	1.84	14	4.98	29
	1,024	9.57	115	3.33	41	5.53	13	9.17	17
	2,048	19.72	142	6.13	42	21.82	13	31.08	21
	4,096	38.51	174	10.60	39	85.68	12	146.50	18
	8,192	133.45	290	24.99	48	354.05	14	t	–
	16,384	164.07	175	47.31	41	1,430.72	13	t	–
	32,768	314.31	191	102.29	49	t	–	t	–
4node-old	32	0.08	30	0.09	20	0.04	7	0.09	10
chem	2	0.04	7	0.03	15	0.03	13	0.04	19
chem-base	2	0.02	6	0.05	14	0.02	13	0.04	22
lands	3	0.02	8	0.02	10	0.02	5	0.03	17
lands-blocks	3	0.01	8	0.02	10	0.02	5	0.03	17
env-aggr	5	0.02	3	0.04	16	0.02	3	0.03	5
env-first	5	0.02	1	0.02	1	0.02	1	0.02	1
env-loose	5	0.01	1	0.02	1	0.02	1	0.02	1
env	15	0.04	3	0.05	15	0.03	3	0.03	5
	1,200	0.34	3	1.73	15	0.48	3	0.76	5
	1,875	0.57	3	2.80	15	0.90	3	1.50	5
	3,780	1.26	3	5.47	15	2.48	3	3.79	5
	5,292	1.96	3	7.67	15	4.51	3	5.89	5
	8,232	3.70	3	12.58	15	10.67	3	12.54	5
	32,928	39.88	3	75.67	15	211.90	3	212.05	5
env-diss-aggr	5	0.03	9	0.05	22	0.03	9	0.03	17
env-diss-first	5	0.02	14	0.04	12	0.02	4	0.03	4
env-diss-loose	5	0.03	15	0.03	5	0.02	4	0.02	4
env-diss	15	0.05	27	0.10	35	0.05	18	0.07	12
	1,200	1.13	24	2.80	35	2.25	18	3.45	19
	1,875	2.50	29	4.49	36	5.52	19	4.52	15
	3,780	5.04	29	8.87	36	20.23	19	8.98	11
	5,292	8.14	34	12.95	38	40.39	17	17.90	13
	8,232	14.21	35	22.49	41	119.88	16	99.19	23
	32,928	79.52	35	112.46	41	t	–	t	–
phone1	1	0.02	1	0.02	1	0.02	1	0.02	1
phone	32,768	48.34	1	48.23	1	73.45	1	73.75	1
stocfor1	1	0.02	6	0.03	6	0.02	2	0.02	2
stocfor2	64	0.10	7	0.12	9	0.18	14	0.23	18
rand0	2,000	10.42	80	6.10	44	30.33	9	93.78	16
	4,000	19.97	69	10.06	32	82.75	8	591.45	14

Table 5 continued

Name	Scen	Benders		Level		TR		RD	
		Time	Iter	Time	Iter	Time	Iter	Time	Iter
	6,000	41.82	108	21.17	51	275.97	9	t	–
	8,000	65.51	127	28.86	50	423.51	9	t	–
	10,000	153.07	230	52.31	71	871.00	10	t	–
rand1	2,000	265.14	391	52.70	74	155.81	12	361.54	17
	4,000	587.22	502	72.30	59	508.18	11	t	–
	6,000	649.58	385	103.00	58	937.74	11	t	–
	8,000	917.24	453	141.81	65	1,801.43	9	t	–
	10,000	1,160.62	430	181.98	63	t	–	t	–
rand2	2,000	1,800.00	818	145.22	65	334.36	12	794.31	17
	4,000	1,616.56	414	170.08	42	813.49	11	t	–
	6,000	t	–	369.35	52	t	–	t	–
	8,000	t	–	369.01	44	t	–	t	–
	10,000	t	–	623.59	52	t	–	t	–
saphir	50	733.37	128	341.86	43	578.87	110	n	–
	100	1,051.89	123	700.44	46	n	–	n	–
	200	t	–	t	–	t	–	n	–
	500	1,109.48	122	608.48	44	1,283.97	99	n	–
	1,000	1,444.17	124	804.11	46	n	–	n	–

t time limit (1,800 s), *m* insufficient memory, *n* numerical difficulties

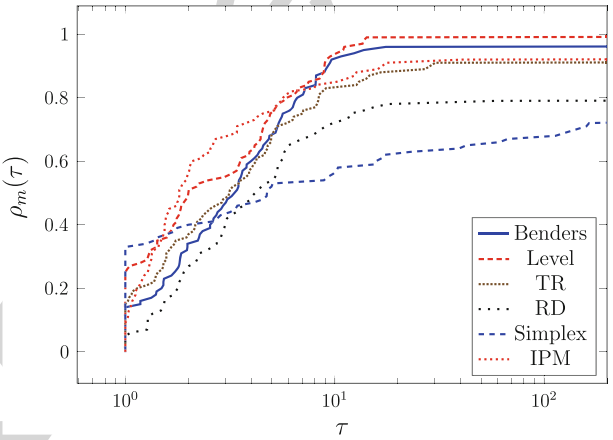

Fig. 1 Performance profiles

The computational results given in the previous section where obtained using the relative stopping tolerance $\epsilon = 10^{-5}$ for the Benders decomposition with and without regularisation by the level method, i.e. the method terminated if $(F^* - F_*)/(|F_*| + 10^{-10}) \leq \epsilon$, where F_* and F^* are, respectively, lower and upper bounds on the value of the objective function. The stopping criteria in the trust region algorithm and

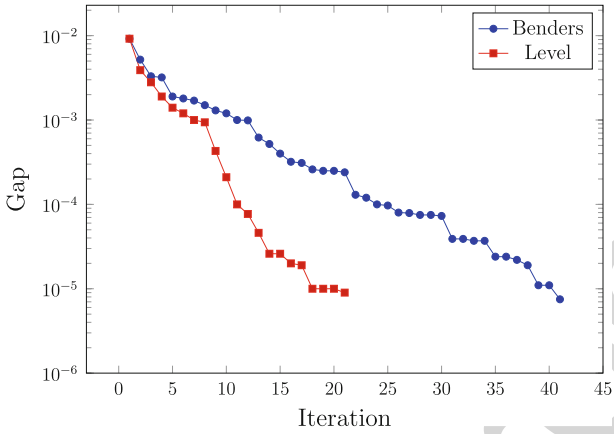

Fig. 2 Gap between lower and upper bounds for storm-1,000 problem

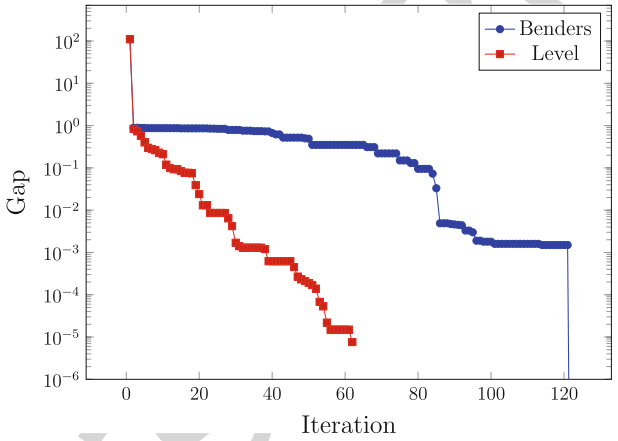

Fig. 3 Gap between lower and upper bounds for 4node-32,768 problem

regularised decomposition are different because these methods do not provide global lower bound. Therefore ϵ was set to a lower value of 10^{-6} which achieves the same precision for most of the problems with the following exceptions that were made to achieve the desirable precision:

- env-diss with 8,232 scenarios: $\epsilon = 10^{-10}$ in RD,
- saphir: $\epsilon = 10^{-10}$ in RD and TR.

For CPLEX barrier optimiser the default complementarity tolerance was used as a stopping criterion.

6 Discussion and conclusion

In this paper we have made a case for continuing research and development of solution algorithms for processing scenario based SP recourse problems in particular two stage

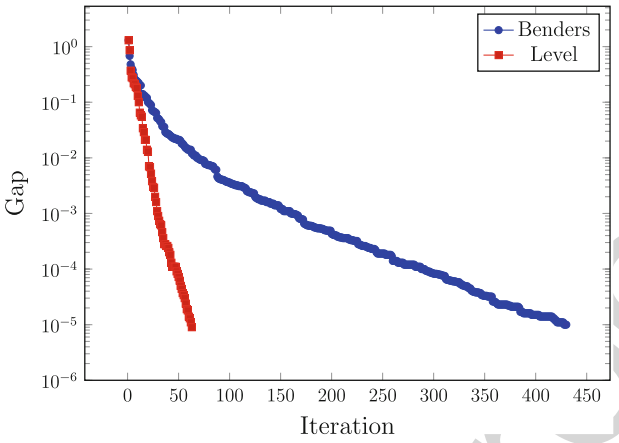

Fig. 4 Gap between lower and upper bounds for rand1-10,000 problem

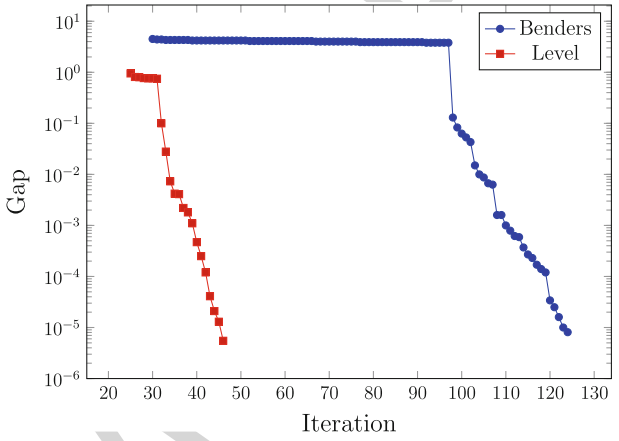

Fig. 5 Gap between lower and upper bounds for saphir-1,000 problem

SPs. Our empirical computational study clearly establishes the need for robust solution methods which can process diverse SP applications in particular as these scale up in size and number of scenarios. We show that simple use of even most powerful hypersparse solvers cannot process many industrial strength models specially, when the model sizes scale up due to multiple scenarios. We also observe that the interior point method outperformed simplex in the majority of cases. All the same IPM applied to the deterministic equivalent problem has a limitation requiring large amount of memory to perform efficiently. In our experiments Benders decomposition performs well, however, through the regularisation by the level method we are able to process very large instances of SP application models. Our empirical study comparing multicut and aggregated regularisation methods reveal that the latter approach scales much better

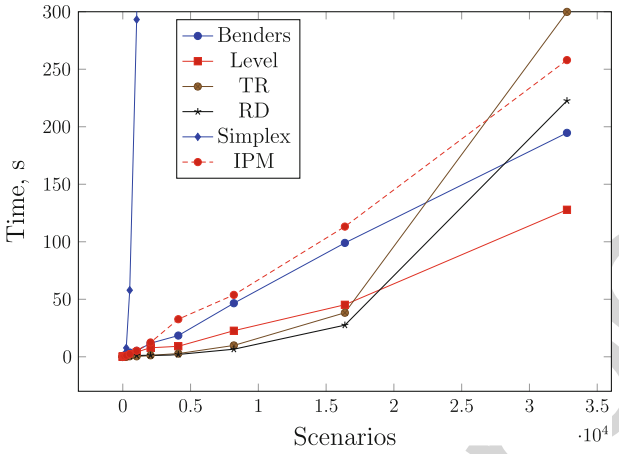

Fig. 6 Time versus the number of scenarios on the 4node problems

then the former hence regularisation through the level method performs well across the entire range of model sizes. We hope to report a similar study for two stage integer stochastic programming benchmark models.

Acknowledgments The research of Victor Zverovich and Dr Eldon Ellison has been supported by OptiRisk Systems. Professor Csaba Fábián's visiting academic position in CARISMA has been supported by OptiRisk Systems, Uxbridge, UK. Also the work of Professor Csaba Fábián has been partially supported by Kecskemét College Grant No 1KU31. These sources of support are gratefully acknowledged. We would also like to thank Professor Jacek Gondzio for providing a copy of the HOPDM solver and the anonymous referees for their constructive comments which resulted in improvements and a considerable extension of the scope of this paper.

References

1. Ariyawansa, K.A., Felt, A.J.: On a new collection of stochastic linear programming test problems. INFORMS J. Comput. **16**(3), 291–299 (2004)
2. Beale, E.M.L.: On minimizing a convex function subject to linear inequalities. J. R. Stat. Soc. B **17**, 173–184 (1955)
3. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. **4**, 238–252 (1962). Re-publised in Comput. Manag. Sci. **2**, 3–19 (2005)
4. Birge, J.R.: Stochastic programming computation and applications: state-of-the-art survey. INFORMS J. Comput. **9**(2), 111–133 (1997)
5. Birge, J.R., Dempster, M.A.H., Gassmann, H.I., Gunn, E.A., King, A.J., Wallace, S.W.: A standard input format for multiperiod stochastic linear programs. COAL Newslett. **17**, 1–19 (1987)
6. Birge, J.R., Louveaux, F.V.: A multicut algorithm for two-stage stochastic linear programs. Eur. J. Oper. Res. **34**, 384–392 (1988)
7. Birge, J.R., Louveaux, F.V.: Introduction to Stochastic Programming. Springer, New York (1997)
8. Colombo, M., Gondzio, J.: Further development of multiple centrality correctors for interior point methods. Comput. Optim. Appl. **41**, 277–305 (2008)
9. Colombo, M., Gondzio, J., Grothey, A.: A warm-start approach for large-scale stochastic linear programs. Math. Program. (2009). doi:10.1007/s10107-009-0290-9
10. Dantzig, G.B.: Linear programming under uncertainty. Manag. Sci. **1**, 197–206 (1955)

11. Dantzig, G.B., Madansky, A.: On the solution of two-stage linear programs under uncertainty. In: Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 165–176. University of California Press, Berkeley (1961)
12. Dantzig, G.B., Wolfe, P.: The decomposition principle for linear programs. Oper. Res. **8**, 101–111 (1960)
13. Di Domenica, N., Lucas, C., Mitra, G., Valente, P.: Scenario generation for stochastic programming and simulation: a modelling perspective. IMA J. Manag. Math. **20**, 1–38 (2009)
14. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
15. Ellison, E.F.D., Mitra, G., Zverovich, V.: FortSP: a stochastic programming solver. OptiRisk Systems. http://www.optirisk-systems.com/manuals/FortspManual.pdf (2010)
16. Fábián, C.I.: Bundle-type methods for inexact data. Central Eur. J. Oper. Res. **8**, 35–55 (2000). [Special issue, Csendes, T., Rapcsák, T. (eds.)]
17. Fábián, C.I., Szőke, Z.: Solving two-stage stochastic programming problems with level decomposition. Comput. Manag. Sci. **4**, 313–353 (2007)
18. Fabozzi, F.J., Focardi, S., Jonas, C.: Trends in quantitative equity management: survey results. Quant. Finance **7**(2), 115–122 (2007)
19. Gassmann, H.: MSLiP: a computer code for the multistage stochastic linear programming problem. Math. Program. **47**, 407–423 (1990)
20. Gassmann, H.I., Wallace, S.W.: Solving linear programs with multiple right-hand sides: pricing and ordering schemes. Ann. Oper. Res. **64**, 237–259 (1996)
21. Gondzio, J.: HOPDM (version 2.12) a fast lp solver based on a primal-dual interior point method. Eur. J. Oper. Res. **85**, 221–225 (1995)
22. Holmes, D.: A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS). http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html (1995)
23. Kall, P., Mayer, J.: On testing SLP codes with SLP-IOR. In: New trends in mathematical programming: homage to Steven Vajda, pp. 115–135. Kluwer, Boston (1998)
24. Kall, P., Mayer, J.: Stochastic linear programming: models, theory, and computation. Springer, International Series in Operations Research and Management Science (2005)
25. Kall, P., Wallace, S.W.: Stochastic Programming. Wiley, Chichester (1994)
26. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Springer, Berlin (1985)
27. König, D., Suhl, L., Koberstein, A.: Optimierung des Gasbezugs im liberalisierten Gasmarkt unter Berücksichtigung von Röhren- und Untertagespeichern. In: Sammelband zur VDI Tagung "Optimierung in der Energiewirtschaft" in Leverkusen (2007)
28. Lemaréchal, C.: Nonsmooth optimization and descent methods. Research Report 78-4, IIASA, Laxenburg, Austria (1978)
29. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**, 111–147 (1995)
30. Linderoth, J., Wright, S.: Decomposition algorithms for stochastic programming on a computational grid. Comput. Optim. Appl. **24**, 207–250 (2003)
31. Mayer, J.: Stochastic Linear Programming Algorithms. Gordon and Breach Science Publishers, Amsterdam (1998)
32. Mitra, G., Di Domenica, N., Birbilis, G., Valente, P.: Stochastic programming and scenario generation within a simulation framework: An information perspective. Decis. Support Syst. **42**, 2197–2218 (2007)
33. Nemirovski, A.: Lectures in Modern Convex Optimization. ISYE, Georgia Institute of Technology (2005)
34. Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM J. Optim. **21**, 517–544 (2011)
35. Prékopa, A.: Stochastic Programming. Kluwer, Dordrecht (1995)
36. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM J. Control Optim. **14**, 877–898 (1976)
37. Ruszczyński, A.: A regularized decomposition method for minimizing a sum of polyhedral functions. Math. Program. **35**, 309–333 (1986)
38. Ruszczyński, A.: Decomposition methods. In: Ruszczyński, A., Shapiro, A. (eds.) Stochastic Programming. Handbooks in Operations Research and Management Science, vol. 10, pp. 141–211. Elsevier, Amsterdam (2003)
39. Ruszczyński, A.: Nonlinear Optimization. Princeton University Press, Princeton (2006)

40. Ruszczyński, A., Shapiro, A. : Stochastic programming models. In: Ruszczyński, A., Shapiro, A. (eds.) Stochastic Programming. Handbooks in Operations Research and Management Science, vol. 10, pp. 1–64. Elsevier, Amsterdam (2003)
41. Ruszczyński, A., Świętanowski, A.: Accelerating the regularized decomposition method for two-stage stochastic linear problems. Eur. J. Oper. Res. **101**, 328–342 (1997)
42. Valente, C., Mitra, G., Sadki, M., Fourer, R.: Extending algebraic modelling languages for stochastic programming. INFORMS J. Comput. **21**(1), 107–122 (2009)
43. Valente, P., Mitra, G., Poojari, C., Ellison, E.F., Di Domenica, N., Mendi, M., Valente, C.: SAMPL/SPInE user manual. OptiRisk Systems. http://www.optirisk-systems.com/manuals/SpineAmplManual.pdf (2008)
44. Van Slyke, R., Wets, R.J.B.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J. Appl. Math. **17**, 638–663 (1969)
45. Wallace, S.W., Ziemba, W.T. (eds.): Applications of Stochastic Programming. Society for Industrial and Applied Mathematic (2005)
46. Wets, R.J.B.: Stochastic programs with fixed recourse: the equivalent deterministic program. SIAM Rev. **16**, 309–339 (1974)