A computational study of a solver system for processing two-stage stochastic linear programming problems *

Victor Zverovich [†‡] Csaba I. Fábián [§¶] Francis Ellison [†‡]
Gautam Mitra [†‡]

December 3, 2010

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 (2005) outlines both the modelling systems for stochastic programming (SP) and many applications in diverse domains.

More recently, Fabozzi et al. (2007) 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

*This is a revised version of the paper which appeared in SPEPS, Volume 2009, Nr. 3.

[†]CARISMA: The Centre for the Analysis of Risk and Optimisation Modelling Applications, School of Information Systems, Computing and Mathematics, Brunel University, UK.

[‡]OptiRisk Systems, Uxbridge, Middlesex, UK.

[§]Institute of Informatics, Kecskemét College, 10 Izsáki út, Kecskemét, 6000, Hungary. E-mail: fabian.csaba@gamf.kefo.hu.

[¶]Department of OR, Loránd Eötvös University, Budapest, Hungary.

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. (2009) for modelling and Mitra et al. (2007) and Di Domenica et al. (2009) 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 Valente et al. (2008, 2009). 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 Dantzig (1955), Beale (1955) and Wets (1974).

A comprehensive treatment of the models and solution methods can be found in Kall and Wallace (1994), Prékopa (1995), Birge and Louveaux (1997), Mayer (1998), Ruszczyński and Shapiro (2003), and Kall and Mayer (2005). Some of these monographs contain extensions of the original model. Colombo et al. (2009) and Gassmann and Wallace (1996) describe computational studies which are based on interior point method and simplex based methods respectively. Birge (1997) 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 section 2 we introduce the model setting of the two stage stochastic programming problem, in section 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 (1986) and then introduce another regularisation approach in some detail, namely, the Level Method (Lemaréchal et al., 1995) adapted for the two-stage stochastic programming problem. Finally we outline the box-constrained trust-region method of Linderoth and Wright (2003).

In section 4 we discuss implementation issues, in section 5 we set out the computational study and in section 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*

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

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:

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

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 objective, $F(\boldsymbol{x}) := \sum_{i=1}^{S} p_i\, q_i(\boldsymbol{x})$, is defined as an expected value and is called the *expected recourse function*. This is a polyhedral convex function with the domain $K := K_1 \cap \ldots \cap K_S$.

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

equivalent problem:

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

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 (1961) observed that the dual of the deterministic equivalent problem fits the prototype for the Dantzig-Wolfe decomposition (1960).

Van Slyke and Wets (1969) proposed a cutting-plane approach for the first-stage problem (2). Their L-Shaped method builds respective cutting-plane models of the feasible domain $K = K_1 \cap \ldots \cap K_S$ 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

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

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 $[W_i^T, I](\boldsymbol{z}, \boldsymbol{\gamma}) = W_i^T \boldsymbol{z} + \boldsymbol{\gamma}$. Given a composite vector $(\boldsymbol{z}, \boldsymbol{\gamma})$ of appropriate dimensions, let $\text{support}(\boldsymbol{z}, \boldsymbol{\gamma})$ denote the set of those column-vectors of the composite matrix $[W_i^T, I]$ that belong to non-zero $(\boldsymbol{z}, \boldsymbol{\gamma})$-components. Using these, let

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

These are finite sets, and 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

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

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

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

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

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

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. The cuts belonging to the model of K are called *feasibility cuts*, and those belonging to the model of F are called *optimality cuts*. Cuts at a given iterate $\widehat{\boldsymbol{x}}$ are generated by solving the dual problems $D_i(\widehat{\boldsymbol{x}})$ $(i = 1, \ldots, S)$. Problems with unbounded objectives yield feasibility cuts, and problems with optimal solutions yield optimality cuts.

In its original form, the L-Shaped method of Van Slyke and Wets (1969) works on the aggregated problem. A *multicut* version that works on the disaggregated problem was proposed by Birge and Louveaux (1988).

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 (1960) applied to the dual of the deterministic equivalent problem (3),

– Benders decomposition (1962) 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 Ruszczyński (2003).

Aggregated vs 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.

Based on the numerical results of Birge and Louveaux (1988) and Gassmann (1990), Birge and Louveaux (1997) 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.

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 Rockafellar (1976), and bundle methods by Lemaréchal (1978) and Kiwiel (1985). 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 (1986) 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 Ruszczyński (2003).

Ruszczyński and Świętanowski (1997) 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, Nemirovskii, and Nesterov (1995). This is a special bundle-type method that uses level sets of the model functions for regularisation. Let us

consider the problem

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

where $X \subset \mathbb{R}^n$ is a convex bounded polyhedron, and f a real-valued convex function, Lipschitzian relative to X. 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 $\underline{F} := \min_{\boldsymbol{x} \in X} \underline{f}(\boldsymbol{x})$ 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

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

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 $\boldsymbol{x} \in \hat{X}$.

Lemaréchal et al. (1995) 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.

Remark 1 *The method in general performs much better than the estimate (8) implies: Fábián and Szőke (2007) solved problems with different settings of the stopping tolerance ϵ, and the number of the required iterations was found to be proportional to* $\log(1/\epsilon)$.

The estimate (8) generally yields bounds too large for practical purposes. We include this estimate because it 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. An interesting finiteness proof that allows for nonbasic solutions is presented in Ruszczyński (2006). 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 *The level method can be implemented for the case when the feasible domain X is not bounded. The theoretical estimate given in (8) is not applicable in this case, but the method works. Finiteness can be proven by applying the arguments set out in remark 1, above.*

Fábián and Szőke (2007) adapted the level method to the solution of two-stage stochastic programming problems, and solved two-stage stochastic programming problems with growing scenario sets. According to the results presented, there is no correlation between the number of the scenarios and the number of the master iterations required (provided that the number of the scenarios is large enough).

Remark 3 *The Level Decomposition (LD) method proposed in Fábián and Szőke (2007) handles feasibility and optimality issues simultaneously, in a unified manner. Moreover the LD framework allows a progressive approximation of the distribution, and approximate solution of the second-stage problems. – This is facilitated by an inexact version Fábián (2000) of the level method. The inexact method uses approximate data to construct a model of the objective function. At the beginning of the procedure, a rough approximation is used, and the accuracy is gradually increased as the optimum is approached. – Numerical results of Fábián and Szőke (2007) 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 solution time of the second-stage problems.*

In this study we use the original (exact) level method hence no progressive approximation is performed. Moreover, only the objective function is regularised, as in the classical regularisation approaches. That is why regularisation does not extend to feasibility issues.

Box-constrained method

The box-constrained trust-region method of Linderoth and Wright (2003) solves the disaggregated problem (5), and uses a special trust-region ap-

proach.

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 (Ellison et al., 2010) 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 pseudocode of each method refer to Ellison et al. 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 (Gondzio, 1995; Colombo and Gondzio, 2008), an implementation of the infeasible primal-dual interior point method.

Table 1: Summary of CPLEX and HOPDM performance

	CPLEX	HOPDM
Average Iterations	29	21
Average Time	56.66	170.50
Solved Problems	87	78

The results summarised in Table 1 show that while it took HOPDM on average less iterations to solve a problem, CPLEX was faster in our benchmarks. This can be explained by the latter being better optimised to the underlying hardware. In particular, CPLEX 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 iterations, 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

$$\begin{array}{rll} \min & \boldsymbol{c^T x} + \vartheta & \\ \text{subject to} & \boldsymbol{x} \in X, \quad \vartheta \in \mathbb{R}, & \\ & \boldsymbol{v_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq 0 & (\boldsymbol{v_i} \in \widehat{V}_i,\ i = 1, \ldots, S), \\ & \sum\limits_{i=1}^{S} p_i \boldsymbol{u_i^T}(\boldsymbol{h_i} - T_i \boldsymbol{x}) \leq \vartheta & \big((\boldsymbol{u_1}, \ldots, \boldsymbol{u_S}) \in \widehat{\mathcal{U}}\big). \end{array} \tag{9}$$

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

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

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 section 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 (1986). 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 (1997):

- 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 $\sigma \leftarrow 2\sigma$,

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:

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

For a more detailed description of the implementation including the pseudo-code please refer to Ellison et al. (2010).

4.5 The trust region method based on the infinity norm

We also implemented the l_∞ trust region L-shaped method of Linderoth and Wright (2003). It operates on the disaggregated problem (5) with additional

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 $(F^* - F^k)/(|F^*| + 10^{-10}) \leq \epsilon$ **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 $(F^* - F^{k+1})/(|F^*| + 10^{-10}) \leq \epsilon$ **then**
    stop
  end if
  solve the projection problem:

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

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

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 Linderoth and Wright (2003) 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. (2010).

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. CPLEX linear and quadratic programming

Table 2: Sources of test problems

	Source	Reference	Comments
1.	POSTS collection	Holmes (1995)	Two-stage problems from the (PO)rtable (S)tochastic programming (T)est (S)et (POSTS)
2.	Slptestset collection	Ariyawansa and Felt (2004)	Two-stage problems from the collection of stochastic LP test problems
3.	Random problems	Kall and Mayer (1998)	Artificial test problems generated with pseudo random stochastic LP problem generator GENSLP
4.	SAMPL problems	König et al. (2007), Valente et al. (2008)	Problems instantiated from the SAPHIR gas portfolio planning model formulated in Stochastic AMPL (SAMPL)

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. (1987).

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 2. Table 3 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.

Table 3: Dimensions of test problems

		Stage 1		Stage 2		Deterministic Equivalent		
Name	Scen	Rows	Cols	Rows	Cols	Rows	Cols	Nonzeros
fxm	6	92	114	238	343	1520	2172	12139
	16	92	114	238	343	3900	5602	31239
fxmev	1	92	114	238	343	330	457	2589
pltexpa	6	62	188	104	272	686	1820	3703
	16	62	188	104	272	1726	4540	9233
stormg2	8	185	121	528	1259	4409	10193	27424
	27	185	121	528	1259	14441	34114	90903
	125	185	121	528	1259	66185	157496	418321
	1000	185	121	528	1259	528185	1259121	3341696
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	4058	5412	16228
assets	100	5	13	5	13	505	1313	2621
	37500	5	13	5	13	187505	487513	975021
4node	1	14	52	74	186	88	238	756
	2	14	52	74	186	162	424	1224
	4	14	52	74	186	310	796	2160
	8	14	52	74	186	606	1540	4032
	16	14	52	74	186	1198	3028	7776
	32	14	52	74	186	2382	6004	15264
	64	14	52	74	186	4750	11956	30240
	128	14	52	74	186	9486	23860	60192
	256	14	52	74	186	18958	47668	120096
	512	14	52	74	186	37902	95284	239904
	1024	14	52	74	186	75790	190516	479520
	2048	14	52	74	186	151566	380980	958752
	4096	14	52	74	186	303118	761908	1917216
	8192	14	52	74	186	606222	1523764	3834144
	16384	14	52	74	186	1212430	3047476	7668000
	32768	14	52	74	186	2424846	6094900	15335712

Table 3: Dimensions of test problems (continued)

		Stage 1		Stage 2		Deterministic Equivalent		
Name	Scen	Rows	Cols	Rows	Cols	Rows	Cols	Nonzeros
4node-base	1	16	52	74	186	90	238	772
	2	16	52	74	186	164	424	1240
	4	16	52	74	186	312	796	2176
	8	16	52	74	186	608	1540	4048
	16	16	52	74	186	1200	3028	7792
	32	16	52	74	186	2384	6004	15280
	64	16	52	74	186	4752	11956	30256
	128	16	52	74	186	9488	23860	60208
	256	16	52	74	186	18960	47668	120112
	512	16	52	74	186	37904	95284	239920
	1024	16	52	74	186	75792	190516	479536
	2048	16	52	74	186	151568	380980	958768
	4096	16	52	74	186	303120	761908	1917232
	8192	16	52	74	186	606224	1523764	3834160
	16384	16	52	74	186	1212432	3047476	7668016
	32768	16	52	74	186	2424848	6094900	15335728
4node-old	32	14	52	74	186	2382	6004	15264
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	2356
	1200	48	49	48	49	57648	58849	177736
	1875	48	49	48	49	90048	91924	277636
	3780	48	49	48	49	181488	185269	559576
	5292	48	49	48	49	254064	259357	783352
	8232	48	49	48	49	395184	403417	1218472
	32928	48	49	48	49	1580592	1613521	4873480
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

Table 3: Dimensions of test problems (continued)

		Stage 1		Stage 2		Deterministic Equivalent		
Name	Scen	Rows	Cols	Rows	Cols	Rows	Cols	Nonzeros
env-diss	15	48	49	48	49	768	784	2356
	1200	48	49	48	49	57648	58849	177736
	1875	48	49	48	49	90048	91924	277636
	3780	48	49	48	49	181488	185269	559576
	5292	48	49	48	49	254064	259357	783352
	8232	48	49	48	49	395184	403417	1218472
	32928	48	49	48	49	1580592	1613521	4873480
phone1	1	1	8	23	85	24	93	309
phone	32768	1	8	23	85	753665	2785288	9863176
stocfor1	1	15	15	102	96	117	111	447
stocfor2	64	15	15	102	96	6543	6159	26907
rand0	2000	50	100	25	50	50050	100100	754501
	4000	50	100	25	50	100050	200100	1508501
	6000	50	100	25	50	150050	300100	2262501
	8000	50	100	25	50	200050	400100	3016501
	10000	50	100	25	50	250050	500100	3770501
rand1	2000	100	200	50	100	100100	200200	3006001
	4000	100	200	50	100	200100	400200	6010001
	6000	100	200	50	100	300100	600200	9014001
	8000	100	200	50	100	400100	800200	12018001
	10000	100	200	50	100	500100	1000200	15022001
rand2	2000	150	300	75	150	150150	300300	6758501
	4000	150	300	75	150	300150	600300	13512501
	6000	150	300	75	150	450150	900300	20266501
	8000	150	300	75	150	600150	1200300	27020501
	10000	150	300	75	150	750150	1500300	33774501
saphir	50	32	53	8678	3924	433932	196253	1136753
	100	32	53	8678	3924	867832	392453	2273403
	200	32	53	8678	3924	1735632	784853	4546703
	500	32	53	8678	3924	4339032	1962053	11366603
	1000	32	53	8678	3924	8678032	3924053	22733103

It should be noted that the problems generated with GENSLP 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.

We refer to the methods using the following abbreviations:

Abbreviation	Full name	Reference
DEP-Simplex	Simplex method applied to the deterministic equivalent problem (DEP)	Section 4.1
DEP-IPM	Interior point method applied to the DEP	Section 4.1
Benders	Benders decomposition	Section 4.2
Level	Benders decomposition with level regularisation	Section 4.3
RD	Regularized decomposition	Section 4.4
TR	Trust region method	Section 4.5

Finally we present the results in the form of performance profiles. The performance profile for a solver is defined by Dolan and Moré (2002) 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:

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

We calculated performance profile of each considered method on the whole set of test problems. These profiles are shown in Figure 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 Figure 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 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 Figure 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.

Table 4: Performance of DEP solution methods and level-regularised decomposition
t - time limit, m - insufficient memory, n - numerical difficulties

		DEP - Simplex		DEP - IPM		Level		Optimal
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Value
fxm	6	0.06	1259	0.05	17	0.15	20	18417.1
	16	0.22	3461	0.13	23	0.15	20	18416.8
fxmev	1	0.01	273	0.01	14	0.13	20	18416.8
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	3649	0.25	28	0.16	20	15535200
	27	0.47	12770	2.27	27	0.31	17	15509000
	125	5.10	70177	8.85	57	0.93	17	15512100
	1000	226.70	753739	137.94	114	6.21	21	15802600
airl-first	25	0.01	162	0.01	9	0.03	17	249102
airl-second	25	0.00	145	0.01	11	0.03	17	269665
airl-randgen	676	0.25	4544	0.05	11	0.22	18	250262
assets	100	0.02	494	0.02	17	0.03	1	-723.839
	37500	1046.85	190774	6.37	24	87.55	2	-695.963

Table 4: Performance of DEP solution methods and level-regularised decomposition (continued)

		DEP - Simplex		DEP - IPM		Level		Optimal
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Value
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	1548	0.11	17	0.15	44	423.013
	32	0.16	2948	0.40	15	0.22	51	423.013
	64	0.72	7185	0.44	17	0.36	54	423.013
	128	2.30	12053	0.50	26	0.47	50	423.013
	256	7.69	31745	1.05	30	0.87	48	425.375
	512	57.89	57200	2.35	30	2.12	51	429.963
	1024	293.19	133318	5.28	32	3.95	53	434.112
	2048	1360.60	285017	12.44	36	7.82	49	441.738
	4096	t	-	32.67	46	9.12	46	446.856
	8192	t	-	53.82	45	22.68	55	446.856
	16384	t	-	113.20	46	45.24	52	446.856
	32768	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	1672	0.11	17	0.10	30	414.688
	32	0.15	3318	0.40	15	0.16	37	416.6
	64	0.49	7745	0.36	13	0.22	33	416.6
	128	1.58	17217	0.33	19	0.35	37	416.6
	256	4.42	36201	0.81	23	0.53	31	417.162
	512	22.44	80941	2.20	29	1.45	37	420.293
	1024	141.91	187231	5.21	32	3.33	41	423.05
	2048	694.89	337082	11.12	32	6.13	42	423.763
	4096	t	-	27.03	37	10.60	39	424.753
	8192	t	-	51.29	40	24.99	48	424.775
	16384	t	-	177.81	73	47.31	41	424.775
	32768	t	-	242.91	48	102.29	49	424.775
4node-old	32	0.20	3645	0.49	18	0.09	20	83094.1
chem	2	0.00	29	0.00	11	0.03	15	-13009.2
chem-base	2	0.00	31	0.00	11	0.05	14	-13009.2
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	20478.7

Table 4: Performance of DEP solution methods and level-regularised decomposition (continued)

		DEP - Simplex		DEP - IPM		Level		Optimal
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Value
env-first	5	0.01	112	0.01	11	0.02	1	19777.4
env-loose	5	0.01	112	0.01	12	0.02	1	19777.4
env	15	0.01	321	0.01	16	0.05	15	22265.3
	1200	1.38	23557	1.44	34	1.73	15	22428.9
	1875	2.90	36567	2.60	34	2.80	15	22447.1
	3780	11.21	73421	7.38	40	5.47	15	22441
	5292	20.28	102757	12.19	42	7.67	15	22438.4
	8232	62.25	318430	m	-	12.58	15	22439.1
	32928	934.38	1294480	m	-	75.67	15	22439.1
env-diss-aggr	5	0.01	131	0.01	9	0.05	22	15963.9
env-diss-first	5	0.01	122	0.01	9	0.04	12	14794.6
env-diss-loose	5	0.01	122	0.01	9	0.03	5	14794.6
env-diss	15	0.01	357	0.02	13	0.10	35	20773.9
	1200	1.96	26158	1.99	50	2.80	35	20808.6
	1875	4.41	40776	3.63	53	4.49	36	20809.3
	3780	16.94	82363	9.32	57	8.87	36	20794.7
	5292	22.37	113894	16.17	66	12.95	38	20788.6
	8232	70.90	318192	m	-	22.49	41	20799.4
	32928	1369.97	1296010	m	-	112.46	41	20799.4
phone1	1	0.00	19	0.01	8	0.02	1	36.9
phone	32768	t	-	50.91	26	48.23	1	36.9
stocfor1	1	0.00	39	0.01	11	0.03	6	-41132
stocfor2	64	0.12	2067	0.08	17	0.12	9	-39772.4
rand0	2000	373.46	73437	9.41	33	6.10	44	162.146
	4000	1603.25	119712	34.28	62	10.06	32	199.032
	6000	t	-	48.84	60	21.17	51	140.275
	8000	t	-	56.89	49	28.86	50	170.318
	10000	t	-	98.51	71	52.31	71	139.129
rand1	2000	t	-	39.97	24	52.70	74	244.159
	4000	t	-	92.71	28	72.30	59	259.346
	6000	t	-	158.24	32	103.00	58	297.563
	8000	t	-	228.68	34	141.81	65	262.451
	10000	t	-	320.10	39	181.98	63	298.638

Table 4: Performance of DEP solution methods and level-regularised decomposition (continued)

		DEP - Simplex		DEP - IPM		Level		Optimal
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Value
rand2	2000	t	-	102.61	22	145.22	65	209.151
	4000	t	-	225.71	24	170.08	42	218.247
	6000	t	-	400.52	28	369.35	52	239.721
	8000	t	-	546.98	29	369.01	44	239.158
	10000	t	-	754.52	32	623.59	52	231.706
saphir	50	269.17	84727	n	-	341.86	43	129505000
	100	685.50	152866	n	-	700.44	46	129058000
	200	t	-	549.45	167	t	-	141473000
	500	t	-	t	-	608.48	44	137871000
	1000	t	-	n	-	804.11	46	133036000

Table 5: Performance of decomposition methods

		Benders		Level		TR		RD	
Name	Scen	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
	1000	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
	37500	87.68	2	87.55	2	172.23	2	114.38	1

Table 5: Performance of decomposition methods (continued)

		Benders		Level		TR		RD	
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Time	Iter
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
	1024	5.14	70	3.95	53	0.42	3	1.23	10
	2048	11.78	83	7.82	49	1.30	4	1.22	5
	4096	18.46	89	9.12	46	2.79	3	2.03	4
	8192	46.56	106	22.68	55	9.87	3	6.59	4
	16384	99.00	110	45.24	52	38.28	3	27.50	4
	32768	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
	512	3.27	84	1.45	37	1.84	14	4.98	29
	1024	9.57	115	3.33	41	5.53	13	9.17	17
	2048	19.72	142	6.13	42	21.82	13	31.08	21
	4096	38.51	174	10.60	39	85.68	12	146.50	18
	8192	133.45	290	24.99	48	354.05	14	t	-
	16384	164.07	175	47.31	41	1430.72	13	t	-
	32768	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

Table 5: Performance of decomposition methods (continued)

		Benders		Level		TR		RD	
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Time	Iter
env-loose	5	0.01	1	0.02	1	0.02	1	0.02	1
	15	0.04	3	0.05	15	0.03	3	0.03	5
	1200	0.34	3	1.73	15	0.48	3	0.76	5
	1875	0.57	3	2.80	15	0.90	3	1.50	5
env	3780	1.26	3	5.47	15	2.48	3	3.79	5
	5292	1.96	3	7.67	15	4.51	3	5.89	5
	8232	3.70	3	12.58	15	10.67	3	12.54	5
	32928	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
	15	0.05	27	0.10	35	0.05	18	0.07	12
	1200	1.13	24	2.80	35	2.25	18	3.45	19
	1875	2.50	29	4.49	36	5.52	19	4.52	15
env-diss	3780	5.04	29	8.87	36	20.23	19	8.98	11
	5292	8.14	34	12.95	38	40.39	17	17.90	13
	8232	14.21	35	22.49	41	119.88	16	99.19	23
	32928	79.52	35	112.46	41	t	-	t	-
phone1	1	0.02	1	0.02	1	0.02	1	0.02	1
phone	32768	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
	2000	10.42	80	6.10	44	30.33	9	93.78	16
	4000	19.97	69	10.06	32	82.75	8	591.45	14
rand0	6000	41.82	108	21.17	51	275.97	9	t	-
	8000	65.51	127	28.86	50	423.51	9	t	-
	10000	153.07	230	52.31	71	871.00	10	t	-
	2000	265.14	391	52.70	74	155.81	12	361.54	17
	4000	587.22	502	72.30	59	508.18	11	t	-
rand1	6000	649.58	385	103.00	58	937.74	11	t	-
	8000	917.24	453	141.81	65	1801.43	9	t	-
	10000	1160.62	430	181.98	63	t	-	t	-
	2000	1800.00	818	145.22	65	334.36	12	794.31	17
	4000	1616.56	414	170.08	42	813.49	11	t	-
rand2	6000	t	-	369.35	52	t	-	t	-
	8000	t	-	369.01	44	t	-	t	-
	10000	t	-	623.59	52	t	-	t	-

Table 5: Performance of decomposition methods (continued)

		Benders		Level		TR		RD	
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Time	Iter
	50	733.37	128	341.86	43	578.87	110	n	-
	100	1051.89	123	700.44	46	n	-	n	-
saphir	200	t	-	t	-	t	-	n	-
	500	1109.48	122	608.48	44	1283.97	99	n	-
	1000	1444.17	124	804.11	46	n	-	n	-

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 Figures 2 – 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.

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

- env-diss with 8232 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.

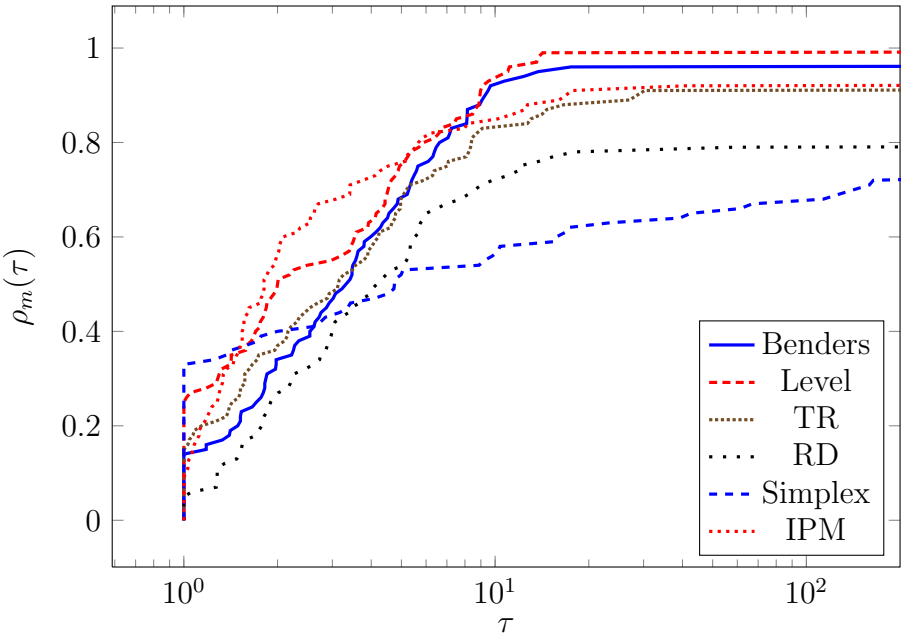

Figure 1: Performance profiles

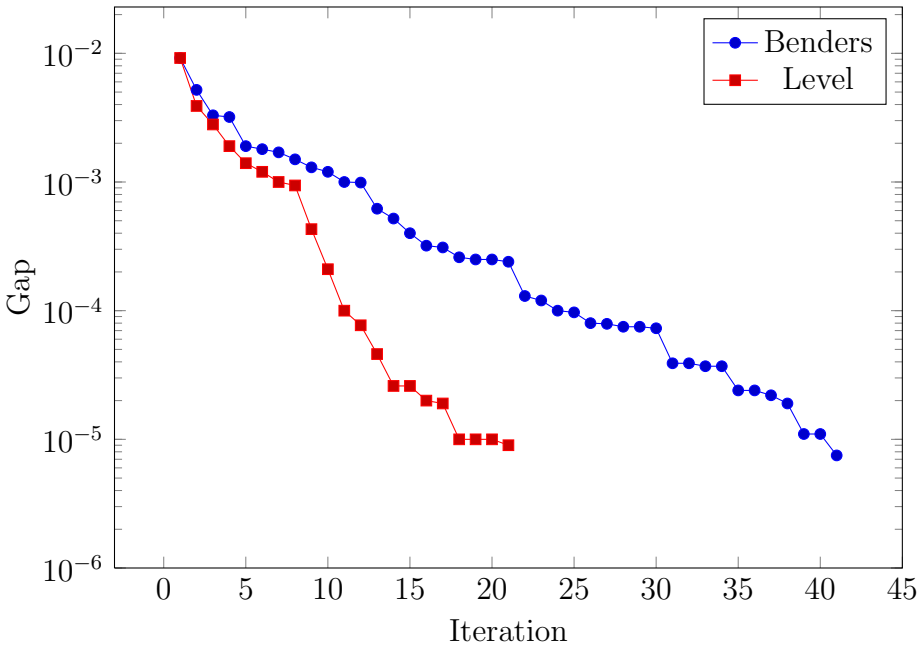

Figure 2: Gap between lower and upper bounds for storm-1000 problem

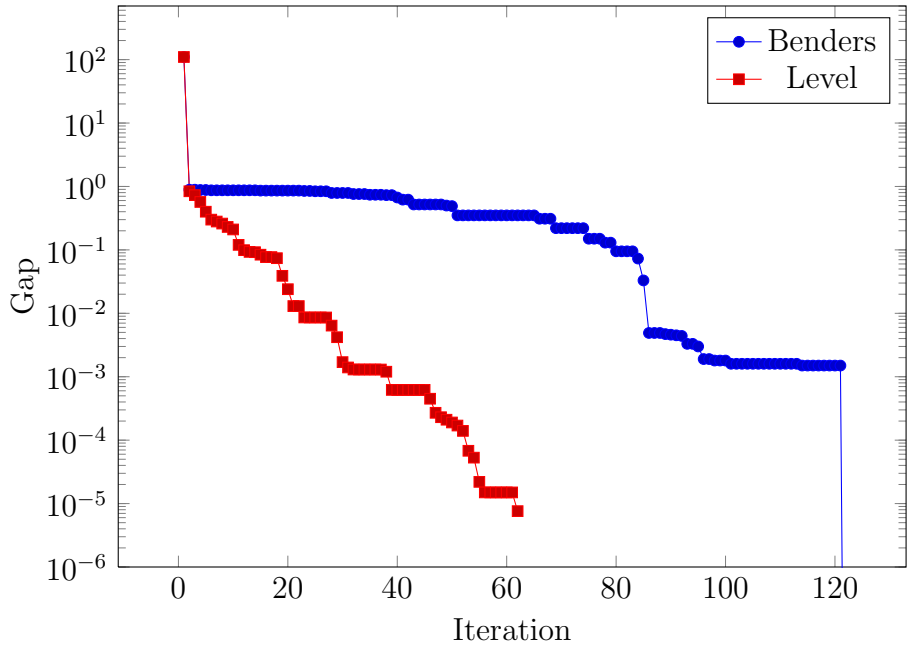

Figure 3: Gap between lower and upper bounds for 4node-32768 problem

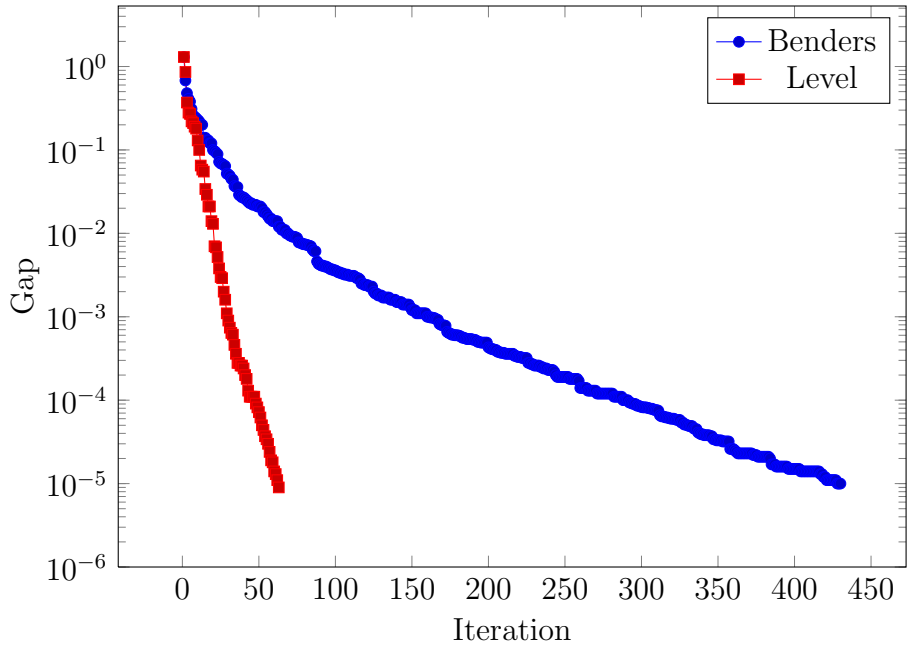

Figure 4: Gap between lower and upper bounds for rand1-10000 problem

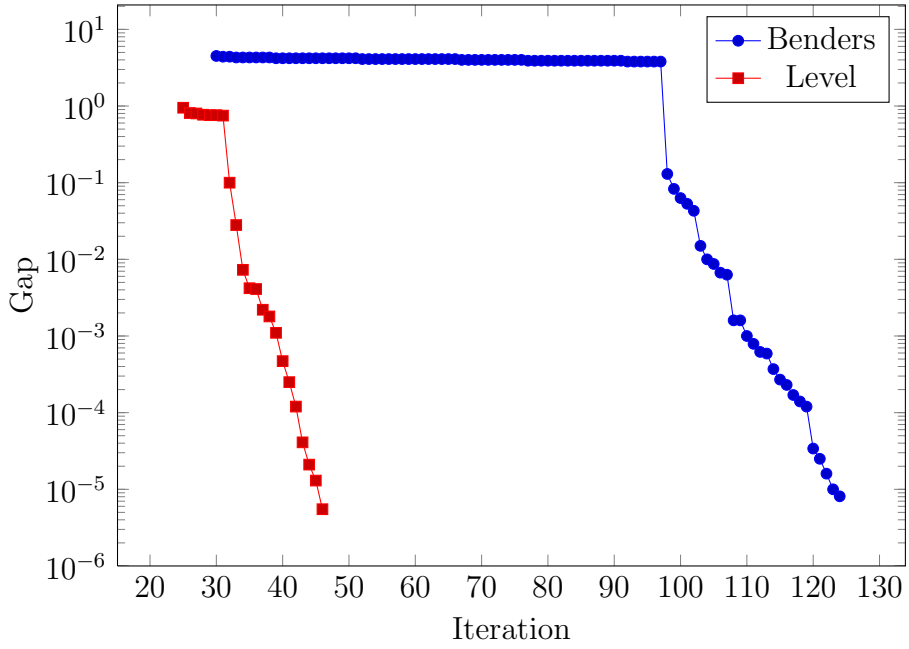

Figure 5: Gap between lower and upper bounds for saphir-1000 problem

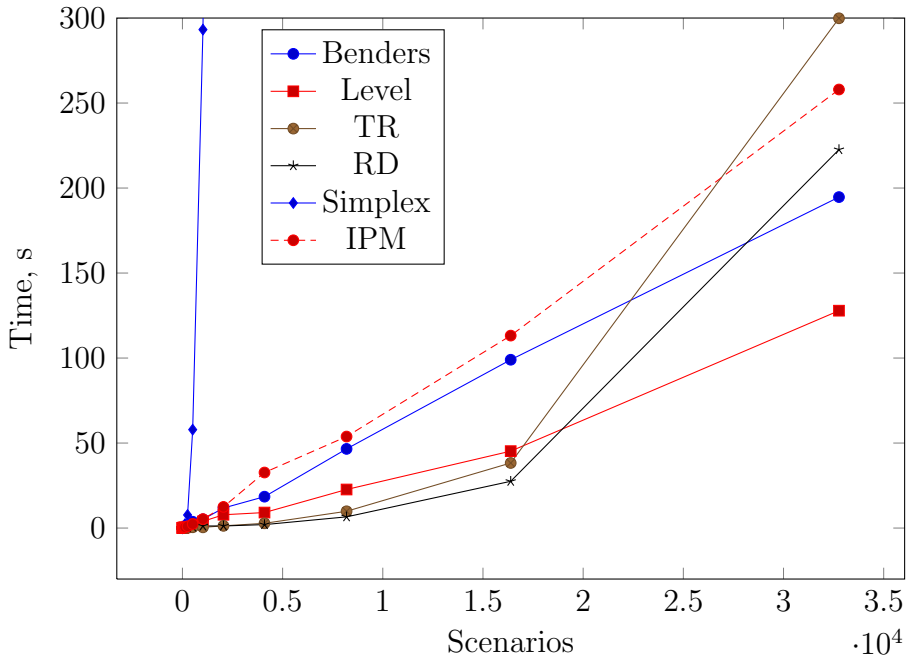

Figure 6: Time vs the number of scenarios on the 4node problems

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 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. 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 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.

Acknowledgements

The research of Victor Zverovich and Dr Francis Ellison has been supported by OptiRisk Systems. Professor Csaba Fábián's visiting academic position in CARISMA has been supported by Optirisk Systems, Uxbridge, UK.

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

Ariyawansa, K. A. and Felt, A. J. (2004). On a new collection of stochastic linear programming test problems. *INFORMS Journal on Computing*, 16(3):291–299.

Beale, E. M. L. (1955). On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society, Series B*, 17:173–184.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables

programming problems. *Numerische Mathematik*, 4:238–252. Re-published in *Computational Management Science 2* (2005), 3–19.

Birge, J. R. (1997). Stochastic programming computation and applications: State-of-the-art survey. *INFORMS Journal on Computing*, 9(2):111–133.

Birge, J. R., Dempster, M. A. H., Gassmann, H. I., Gunn, E. A., King, A. J., and Wallace, S. W. (1987). A standard input format for multiperiod stochastic linear programs. *COAL Newsletter*, 17:1–19.

Birge, J. R. and Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34:384–392.

Birge, J. R. and Louveaux, F. V. (1997). *Introduction to Stochastic Programming*. Springer-Verlag, New York.

Colombo, M. and Gondzio, J. (2008). Further development of multiple centrality correctors for interior point methods. *Computational Optimization and Applications*, 41:277–305.

Colombo, M., Gondzio, J., and Grothey, A. (2009). A warm-start approach for large-scale stochastic linear programs. *Mathematical Programming*. DOI:10.1007/s10107-009-0290-9.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, 1:197–206.

Dantzig, G. B. and Madansky, A. (1961). On the solution of two-stage linear programs under uncertainty. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 165–176. University of California Press, Berkeley.

Dantzig, G. B. and Wolfe, P. (1960). The decomposition principle for linear programs. *Operations Research*, 8:101–111.

Di Domenica, N., Lucas, C., Mitra, G., and Valente, P. (2009). Scenario generation for stochastic programming and simulation: a modelling perspective. *IMA Journal of Management Mathematics*, 20:1–38.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Ellison, F., Mitra, G., and Zverovich, V. (2010). *FortSP: A Stochastic Programming Solver*. OptiRisk Systems. `http://www.optirisk-systems.com/manuals/FortspManual.pdf`.

Fábián, C. I. (2000). Bundle-type methods for inexact data. *Central European Journal of Operations Research*, 8:35–55. Special issue, T. Csendes and T. Rapcsák, eds.

Fábián, C. I. and Szőke, Z. (2007). Solving two-stage stochastic programming problems with level decomposition. *Computational Management Science*, 4:313–353.

Fabozzi, F. J., Focardi, S., and Jonas, C. (2007). Trends in quantitative equity management: survey results. *Quantitative Finance*, 7(2):115–122.

Gassmann, H. (1990). MSLiP: a computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47:407–423.

Gassmann, H. I. and Wallace, S. W. (1996). Solving linear programs with multiple right-hand sides: Pricing and ordering schemes. *Annals of Operations Research*, 64:237–259.

Gondzio, J. (1995). Hopdm (version 2.12) a fast lp solver based on a primal-dual interior point method. *European Journal of Operational Research*, 85:221–225.

Holmes, D. (1995). A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS). `http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html`.

Kall, P. and Mayer, J. (1998). On testing SLP codes with SLP-IOR. In *New Trends in Mathematical Programming: Homage to Steven Vajda*, pages 115–135. Kluwer Academic Publishers.

Kall, P. and Mayer, J. (2005). *Stochastic Linear Programming: Models, Theory, and Computation*. Springer, International Series in Operations Research and Management Science.

Kall, P. and Wallace, S. W. (1994). *Stochastic Programming*. John Wiley & Sons, Chichester.

Kiwiel, K. C. (1985). *Methods of descent for nondifferentiable optimization*. Springer-Verlag, Berlin, New York.

König, D., Suhl, L., and Koberstein, A. (2007). Optimierung des Gasbezugs im liberalisierten Gasmarkt unter Berücksichtigung von Röhren- und Untertagespeichern. In *Sammelband zur VDI Tagung "Optimierung in der Energiewirtschaft" in Leverkusen*.

Lemaréchal, C. (1978). Nonsmooth optimization and descent methods. *Research Report* 78-4, IIASA, Laxenburg, Austria.

Lemaréchal, C., Nemirovskii, A., and Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, 69:111–147.

Linderoth, J. and Wright, S. (2003). Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24:207–250.

Mayer, J. (1998). *Stochastic Linear Programming Algorithms*. Gordon and Breach Science Publishers, Amsterdam.

Mitra, G., Di Domenica, N., Birbilis, G., and Valente, P. (2007). Stochastic programming and scenario generation within a simulation framework: An information perspective. *Decision Support Systems*, 42:2197–2218.

Prékopa, A. (1995). *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht.

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14:877–898.

Ruszczyński, A. (1986). A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35:309–333.

Ruszczyński, A. (2003). Decomposition methods. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming, Handbooks in Operations Research and Management Science*, volume 10, pages 141–211. Elsevier, Amsterdam.

Ruszczyński, A. (2006). *Nonlinear Optimization*. Princeton University Press.

Ruszczyński, A. and Shapiro, A. (2003). Stochastic programming models. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming, Handbooks in Operations Research and Management Science*, volume 10, pages 1–64. Elsevier, Amsterdam.

Ruszczyński, A. and Świętanowski, A. (1997). Accelerating the regularized decomposition method for two-stage stochastic linear problems. *European Journal of Operational Research*, 101:328–342.

Valente, C., Mitra, G., Sadki, M., and Fourer, R. (2009). Extending algebraic modelling languages for stochastic programming. *Informs Journal on Computing*, 21(1):107–122.

Valente, P., Mitra, G., Poojari, C., Ellison, E. F., Di Domenica, N., Mendi, M., and Valente, C. (2008). *SAMPL/SPInE User Manual*. OptiRisk Systems. `http://www.optirisk-systems.com/manuals/SpineAmplManual.pdf`.

Van Slyke, R. and Wets, R. J. B. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663.

Wallace, S. W. and Ziemba, W. T., editors (2005). *Applications of Stochastic Programming*. Society for Industrial and Applied Mathematic.

Wets, R. J. B. (1974). Stochastic programs with fixed recourse: The equivalent deterministic program. *SIAM Review*, 16:309–339.