ON THE NUMERICAL SOLUTION OF STOCHASTIC OPTIMIZATION PROBLEMS

J. Mayer[1]

[1] *Institutefor Operations Research, University ofZurich, Zurich, Switzerland, mayer@ior.unizh.ch*

Abstract We introduce the stochastic linear programming (SLP) model classes, which will be considered in this paper, on the basis of a small–scale linear programming problem. The solutions for the various problem formulations are discussed in a comparative fashion. We point out the need for model and solution analysis. Subsequently, we outline the basic ideas of selected major algorithms for two classes of SLP problems: two–stage recourse problems and problems with chance constraints. Finally, we illustrate the computational behavior of two algorithms for large–scale SLP problems.

keywords: stochastic linear programming, numerical methods.

1. SLP problem formulations

Our starting point is a simple deterministic linear programming (LP) production problem which serves for illustrating various model formulations in stochastic linear programming (SLP). Two kinds of raw materials are used for producing a single good, and we consider a single planning period. The LP–formulation for minimizing costs reads as

$$\begin{array}{llrcrl} \textit{Costs:} & z = & 2\,x_1 & + & 3\,x_2 & \rightarrow \min \\ \textit{Capacity:} & & x_1 & + & x_2 & \leq 100 \\ \textit{Demand:} & & a_1\,x_1 & + & a_2\,x_2 & \geq b \\ & & x_1, & & x_2 & \geq 0 \end{array} \tag{1}$$

where x_1 and x_2 denote the amounts of raw materials to be used for the production; these are our decision variables. The overall storage capacity for the raw materials is 100, and the prices are 2 and 3 in some monetary unit, respectively. b denotes the demand for the product whereas a_1 and a_2 stand for the productivity factors of the two raw materials, respectively.

a_1, a_2, and b will be considered as parameters. Choosing $a_1 = 5$, $a_2 = 8$, and $b = 640$, we get the solution $x_1^* = 0$, $x_2^* = 80$, $z^* = 240$. Note that in this solution the storage capacity is not fully utilized and only the second raw material is used for the production.

In the sequel, we will assume that the parameters a_1, a_2, and b are stochastically independent, normally distributed random variables with the probability distribution not depending on x_1 and x_2: $a_1 \sim \mathcal{N}(5, 0.2)$, $a_2 \sim \mathcal{N}(8, 0.6)$, and $b \sim \mathcal{N}(640, 14)$. The question arises, how to interpret (1) under such circumstances.

1.1 First idea: the expected value problem

The simplest idea is to replace the random parameters by their expected values $\mathbf{E}[a_1] = 5$, $\mathbf{E}[a_2] = 8$, $\mathbf{E}[b] = 625$ and solve the resulting deterministic LP. In our case, this yields the solution discussed in the previous section.

A clear drawback of this approach is that we get the same solution for all probability distributions having the same expected value. Unfortunately, due to its simplicity, the expected value problem is widely used in practice as a substitute of the stochastic problem. As we will see later, the expected value solution behaves in our case extremely badly, when taking the true stochastic nature of the problem into the account.

1.2 A robust interpretation: "fat" solutions

The next idea is to take problem (1) as it stands, with each of the realizations generating a constraint. This idea is due to Madansky who termed the solution obtained this way as "fat solution". Having continuous distributions with an unbounded support, we arrive at a problem with infinitely many constraints, and have no chance to get a feasible solution. Thus, as a next step, let us replace the original distribution with an empirical one. Discretizing the distribution with $(a_1, a_2, b) \sim (9 \times 9 \times 9) = 729$ and with $\sim (5 \times 5 \times 5) = 125$ realizations, the problem turns out to be still infeasible. Finally, taking the rather crude discretization with $(a_1, a_2, b) \sim (3 \times 3 \times 3) = 27$ realizations we get an optimal solution. This illustrates the main drawback of the approach: typically we have no feasible solutions for the reformulated problems. Another drawback is that instead of the probability distribution only the support of the distribution enters the model; we obtain the same solution for any two probability distributions having the same support.

1.3 Chance constraints

Regarding the stochastic demand constraint in (1), the next idea is to evaluate the quality of a decision by computing the probability of the event that the constraint inequality holds. Prescribing the probability on a high level leads to chance constrained problems (or probabilistic constrained problems). In our case, we get a chance constrained problem by replacing the demand constraint in (1) by the probability constraint

$$\mathbf{P}(a_1 x_1 + a_2 x_2 \geq b) \geq \alpha \tag{2}$$

with α being a high probability level. We have solved our example with probability levels $\alpha = 0.95$ and $\alpha = 0.99$. Note that positive values of the quantity $\zeta(x_1, x_2) := b - a_1 x_1 - a_2 x_2$ represent unfulfilled demands. We interpret these as *losses*. Thus our chance constrained model provides a solution, for which the probability of a loss is small $(1 - \alpha)$. Nevertheless, losses my occur, and for the case when losses occur, chance constrained models have no built–in facilities for controlling the size of a loss.

1.4 Integrated chance constraints and CVaR constraints

Constraining the size of the expected loss leads to models with integrated chance constraints. In our case we obtain a model of this type by replacing the demand constraint in (1) by the constraint

$$\mathbf{E}[(b - a_1 x_1 + a_2 x_2)^+] \leq \gamma_{icc}$$

with γ_{icc} being a maximum tolerable loss and $u^+ = \max\{0, u\}$ for any real number u. In our computations, we have chosen $\gamma_{icc} = 5$ and have discretized the probability distribution with $(a_1, a_2, b) \sim (10 \times 10 \times 10) = 1000$ realizations.

A related idea gaining increasing importance in financial applications, is based on conditional value–at–risk (CVaR). In our continuously distributed case, the idea can be interpreted as constraining the expected loss, given that it exceeds the α–quantile of the loss, $VaR_\alpha(\zeta(x_1, x_2))$. In our example, the demand constraint in (1) is substituted by the constraint

$$\mathbf{E}[\zeta(x_1, x_2) \mid \zeta(x_1, x_2) \geq VaR_\alpha(\zeta(x_1, x_2))] \leq \gamma_{cvar}$$

with γ_{cvar} being a maximum tolerable CVaR value. Although in our normally distributed case the problem can equivalently be formulated as a nonlinear programming problem, we have discretized the probability distribution as before and took $\gamma_{cvar} = 5$ in our computations. We have chosen the probability level as $\alpha = 0.95$.

1.5 Two–stage recourse model

We introduce penalty costs both for $\zeta(x_1, x_2) = b - a_1 x_1 - a_2 x_2 < 0$ and for $\zeta(x_1, x_2) > 0$ and consider the random variable
$Q(x_1, x_2; a_1, a_2, b) =$

$$= \min \left. \begin{array}{cccl} 7y_1 & + & 2y_2 & \\ y_1 & - & y_2 & = b - a_1 x_1 - a_2 x_2 \\ y_1, & & y_2 & \geq 0 \end{array} \right\} \tag{3}$$

where the penalty costs of 7 arise if the demand is not fulfilled and the costs of 2 stand for overproduction. The idea is to evaluate solutions via the expected overall costs. The two–stage recourse model arises from (1) by augmenting the objective function by the expected costs, leading to

$$2x_1 + 3x_2 + \mathbf{E}[\,Q(x_1, x_2; a_1, a_2, b)],$$

and by dropping the demand constraint. Note that we still have a single time period, say $[0, T]$, but a two–stage decision. At time $t = 0$ we have to decide on x_1 and x_2, taking into account the expected costs of the recourse actions at $t = T$ (represented by the variables y_1 and y_2). The latter clearly depend on x_1, x_2, and also on the distribution of the random entries.

1.6 Wait–and–See solution

This means solving

$$\left.\begin{array}{lllll} \mathbf{E}[\ \min & 2\,x_1 & + & 3\,x_2 & +Q(x_1, x_2; a_1, a_2, b)\ ] \\ & x_1 & + & x_2 & \leq 100 \\ & x_1, & & x_2 & \geq 0 \end{array}\right\} \qquad (4)$$

and amounts in computing the optimal objective values separately for the realizations and computing subsequently the expected value. In our computations, we took a discrete approximation with $10 \times 10 \times 10 = 1000$ realizations. In general, for the different realizations different solution vectors are obtained. One might get the idea to construct a solution by taking the expected value of the solutions for the separate realizations. As we will see, our example indicates that this is usually not a good idea.

1.7 Computational results, outlook on algorithms

Table 1.7 displays the results obtained by solving the SLP–variants of the production problem. The rows correspond to the expected value problem, to the fat formulation, to the chance constrained problem (with probability levels 0.95 and 0.99), to integrated chance constraint, to CVaR constraint, to the two–stage recourse problem, and to the wait–and–see problem, respectively. The second and third columns display the components of the optimal solution; the fourth column shows the optimal objective value of the corresponding SLP problem.

The column headed by **P** shows the probabilities (2) computed for the optimal solutions obtained from the various SLP models. The last column displays the overall expected costs in the two–stage recourse problem, when fixing the first–stage variables according to the optimal solutions from the second and third column.

Comparing the solutions obtained from the various approaches, we observe

	x_1^*	x_2^*	z^*	**P**	**R**–cost
Exp	0.00	80.00	240.00	0.49	378.71
Fat	0.00	94.05	282.14	0.98	285.82
CC95	28.24	71.76	271.76	0.95	277.63
CC99	9.63	90.37	290.38	0.99	291.37
ICC	44.26	55.74	255.74	0.77	290.73
CVar	21.66	78.34	278.33	0.97	281.49
RS	32.59	67.41	277.08	0.93	277.08
WSS	7.50	75.23	241.10	0.69	375.19

Table 1. Computational results for the example

a great diversity. The expected value solution and the fat solution, for instance, suggest a production plan, solely based on the second raw material. In addition, for these solutions the storage capacity is not fully utilized. Contrary to this, the ICC solution proposes a balanced usage of the two raw materials. The question arises: Which of these is the "true" solution of our stochastic problem? Clearly none of them can be identified as ultimately best; the proper choice depends on the modeling attitude and also on available solvers (implementations of solution algorithms).

According to Richard W. Hamming, "the purpose of computing is insight, not numbers." In our case, we have built and solved several SLP problems corresponding to different modeling paradigms and based on the same initial deterministic LP model and the same probability distribution. The last two columns in Table 1.7 display an evaluation of the solutions obtained, based on two quality measures: the probability that the demand will be fulfilled and the overall expected costs. According to this, the expected value solution is by far the worst, having the lowest probability and highest costs. Almost as worse is the solution obtained form the naïve application of the wait–and–see approach, with averaged solutions. The proper choice clearly depends on the risk–cost attitude of the modeler. Assuming a modeler who places approximately equal weights on risk and costs, a good solution appears to be the C95 solution.

When working with a single modeling paradigm, analysis of the model instance and the solution should be part of the modeling process. As an example for model instance analysis, let us consider the two–stage recourse formulation of our example. We may compute the expected value of perfect information (EVPI) and the value of stochastic solution (VSS) according to

$$EVPI := z^R - z^W = 35.96 \quad \text{and} \quad VSS := z^V - z^R = 101.24$$

where z^W and z^R are the optimal objective values of the wait-and-see problem and the two–stage recourse problem, respectively. z^V is the objective value of the two–stage problem with x fixed as a solution of the expected value problem. These quantities are interpreted as valuing the effort of building a stochastic model, instead of taking the expected value problem, for instance. Loosely speaking, $EVPI$ and VSS indicate a "degree of stochasticity" of the model instances. For details see [2] or [11]. According to these measures, our example counts as highly stochastic.

For SLP problems, the main numerical difficulties have their roots in the expected values and probabilities involved in the model formulations. In general, computing them amounts in computing multivariate integrals. Regarding expectations, the main solution approaches are based on approximating the probability distribution by finite discrete distributions. Thus, the integrals reduce to sums, leading to (typically large–scale) LP problems. For chance constraints the integrals are evaluated by Monte–Carlo methods, which is time–consuming and provides results with a relatively low accuracy.

In the next sections we will outline the basic ideas of the algorithms used in our computations. We will not discuss algorithms for integrated chance constraints and for CVaR constraints. For these methods see [15], [16], as well as [11]. Introductory textbooks for SLP algorithms are [2] and [14]. For algorithms discussed in a detailed fashion see the books [5], [7], [11], [18], [19], and [22]. For comparative computational results involving several algorithms see [13] and the references therein.

2. Algorithms: chance constraints

The general problem formulation is

$$\left.\begin{array}{lccc} \min & c^{\mathrm{T}}x & & \\ & \mathbf{P}(\,T(\xi)x \ge h(\xi)\,) & \ge & \alpha \\ & x & \in & \mathcal{B} \end{array}\right\} \qquad (5)$$

with α being a high probability level and $\mathcal{B} = \{x \mid Ax = b, x \ge 0\}$. The two basic classes of chance constraints are:

Separate chance constraints: The probability applies to a single inequality ($T(\xi)$ has a single row). For some distributions, including the normal, and sufficiently high probability levels, reformulations into convex NLP problems in algebraic terms exist, see [11] or [22].

Joint chance constraints: The probability applies to a vector inequality ($T(\xi)$ may involve several rows). If only the right–hand–side (RHS) is stochastic, the above problem is a convex programming problem for some distributions, including the normal, see [11] or [22].

In the special case, when only the RHS is stochastic, by taking $T(\xi) \equiv T$

and $h(\xi) \equiv \xi$, (5) can be written as

$$\left.\begin{array}{rcl} \min & c^{\mathrm{T}}x & \\ F(Tx) & \geq & \alpha \\ x & \in & \mathcal{B} \end{array}\right\} \tag{6}$$

where we utilized $\mathbf{P}(Tx \geq \xi) = F(Tx)$, with F being the probability distribution function of ξ. In the sequel, we assume that ξ has a multivariate normal distribution. In this case F turns out to be a logconcave function (see [22]) and (6) becomes a convex programming problem. Nevertheless, the problem turns out to be difficult to solve numerically.

On the one hand, the computation of F and its gradient ∇F is a numerically difficult problem, which can only be carried out via Monte–Carlo integration methods in higher dimensions. Therefore, as far as possible, algorithms are utilizing cheaply computable Boole–Bonferroni–type bounds. On the other hand, the graph of F, except for a relatively small non–convex region, consists of extremely flat regions with practically vanishing gradients. Therefore, algorithms utilize Slater–points (feasible points x with $F(Tx) > \alpha$) as navigation aids in the iteration process.

A detailed discussion of the numerical issues can be found in [20]. For the Monte–Carlo techniques applied for the multivariate normal distribution function and for the techniques for computing Boole–Bonferroni bounds see [11], [22], and the references therein.

The algorithms for jointly chance–constrained problems are constructed in the following way: a general nonlinear programming algorithm is taken and subsequently specialized to the problem structure. As an example let us consider the central cutting plane method of Elzinga and Moore [3], endowed with a moving Slater–point by Mayer [19]. Figure 2 displays two iterations of the method.

On the left–hand-side of the figure, the feasible domain of (6) is indicated by

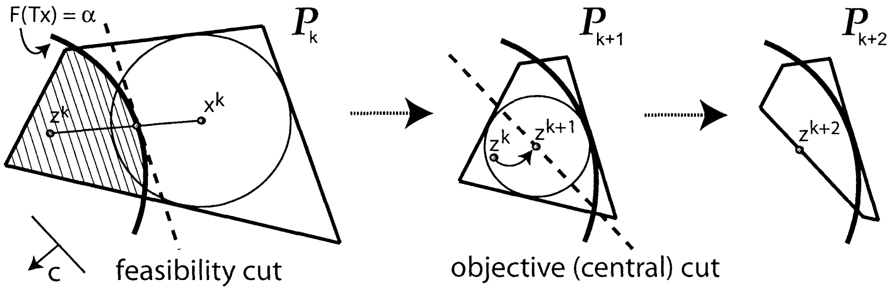

Figure 1. The central cutting plane method

the shaded region. P_k is a convex polyhedron, containing the feasible domain and z^k is the current Slater–point. First the center x^k of the largest hypersphere, inscribed into P_k is computed, which can be carried out by solving an LP problem. The center x^k lies in this case outside of the feasible region. Subsequently the intersection of the boundary of the feasible domain and of the straight line segment joining x^k and the Slater–point z^k is computed. For this computation Boole–Bonferroni bounds are utilized. Applying a *feasibility cut* via a supporting hyperplane leads to the convex polyhedron P_{k+1}, still containing the feasible domain, as it can be seen in the central part of the figure.

In P_{k+1}, the center of the largest inscribed hypersphere belongs to the feasible domain. In this case, the center becomes the new Slater–point z^{k+1} and an *objective cut* is carried out, where the cutting plane passes through z^{k+1} and is parallel to the contour–hyperplanes of the objective function. The objective cut cuts off a portion of the feasible domain, nevertheless, the optimal solution still belongs to the reduced convex polyhedron P_{k+2}, shown in the right–hand–side of the figure.

For details concerning this method, including a theoretical discussion, see [19].

3. Algorithms: two–stage recourse problems, empirical distribution

The general formulation of two–stage fixed recourse problems is

$$\left.\begin{array}{rl} \min & c^{\mathrm{T}}x + \mathbf{E}[\,Q(x,\xi)\,] \\ & x \in \mathcal{B} \end{array}\right\} \tag{7}$$

where $\mathcal{B} = \{x \mid Ax = b, x \geq 0\}$ and the *recourse subproblem* is

$$\left.\begin{array}{rcrcl} Q(x,\xi) & = & \min & q^{\mathrm{T}}y & \\ & & Wy & \geq & h(\xi) - T(\xi)x \\ & & y & \geq & 0. \end{array}\right\} \tag{8}$$

where W is called the recourse matrix. Due to the fact that W is not stochastic, (7) belongs to the class of *fixed recourse problems.* Problem (7) is called a *complete recourse* problem, if the recourse subproblem (8) has feasible solutions for any x and ξ. The problem counts as having *simple recourse* if $W = I$ and $T(\xi) \equiv T^0$ hold. The random entries in the model arrays can frequently be modeled as

$$T(\xi) = T^0 + \sum_{k=1}^{r} T^k \xi_k, \quad h(\xi) = h^0 + \sum_{k=1}^{r} h^k \xi_k,$$

for instance, via principal component analysis, where T^k, h^k are deterministic arrays. $\xi_1, \ldots, \xi_r$ are in many cases stochastically independent.

Now we assume that ξ has an empirical distribution with L realizations (scenarios) $\hat{\xi}^k$ and corresponding probabilities p_k, $k = 1, \ldots, L$. Let, furthermore, $T^k = T(\hat{\xi}^k)$, $h^k = h(\hat{\xi}^k)$, $k = 1, \ldots, L$.

In this case the problem can equivalently be formulated as a deterministic LP problem, having the structure as displayed in Figure 3.

Naïve view: the discretely distributed case is easy to handle numerically;

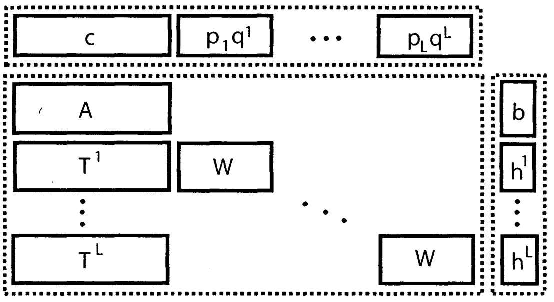

Figure 2. Dual block–angular structure of the equivalent LP

just solve this LP by readily available general–purpose LP solvers. To see the difficulty, just take 10 independent random variables, each with 10 realizations. The number of diagonal blocks will be $L = 10^{10}$. Thus, also in the discretely distributed case, ideas are needed.

In fact, a first idea is to utilize the special structure of the LP. There are two main classes of algorithm in this category.

The first class consists of decomposition methods, the most widely used algorithms will be discussed in the next section.

Interior point methods belong to the second class, where the algorithm of Mészáros [21] turned out to be one of the best in our numerical experiments.

3.1 Decomposition methods

These methods are based on the following basic observation: the expected value of the recourse function

$$f(x) := \mathbf{E}[\,Q(x,\xi)\,] = p_1\,Q(x,\xi^1) + \ldots, p_L\,Q(x,\xi^L)$$

is a piecewise linear convex function.

The basic decomposition method is due to Benders [1]. Its specialized version to SLP–problems, called L–shaped method, has been developed by Van Slyke and Wets [25]. The main idea is to apply the cutting plane method to the epigraph of f. Having x^ν as the current iterate, proceed as follows:
$\mapsto$ Compute $f(x^\nu)$ by solving L recourse problems (8) via the simplex method. Fortunately, utilizing the dual solutions, this also provides a supporting cutting

hyperplane.
$\mapsto$ Check the optimality criterion $f(x^\nu) \leq \theta^\nu + \varepsilon$.
$\mapsto$ If the algorithm does not stop, apply a cut. Technically, the cuts are collected as constraints in the *relaxed master problem*

$$\left.\begin{array}{rrrll} \theta^\nu := \min & c^{\mathrm{T}}x & +z & & \\ & D_k x & -z & \leq d_k, & k = 1, \dots, \nu \\ & x & \in & \mathcal{B} & \end{array}\right\}$$

$\mapsto$ Solve the current relaxed master problem to obtain $x^{\nu+1}$.

This approach has, however, some drawbacks. On the one hand, the method produces large steps in the beginning phase, even with a nearly optimal starting solution. On the other hand, there is no reliable strategy for dropping redundant cuts.

Both of these shortcomings are eliminated in the regularized decomposition method of Ruszczyński [23]. The main idea is to add a regularizing term to the objective of the relaxed master problem:

$$\theta^\nu := \min c^{\mathrm{T}}x + \lambda \|x - \hat{x}^\nu\|^2 + z$$

where $\hat{x}_\nu$ is the current *candidate solution* and $\lambda > 0$ holds. The candidate solution is changed, only if the solution $f(x^\nu)$ is sufficiently smaller than $f(\hat{x}^\nu)$. Additionally, it turns out that it is sufficient to keep at most $n + L$ cuts.

4. Algorithms: two–stage recourse problems, general distributions

Decomposition methods certainly help to solve problems with a large number of realizations. It is still open, however, what to do if in the discretely distributed case we have, for instance, $L = 10^{10}$ joint realizations. A further problem is, what to do if ξ has a continuous distribution?

We will consider the main ideas of three basic approaches in the subsequent sections. An additional general approach is based on stochastic quasi–gradients; for these methods see Marti [18].

4.1 Successive discrete approximation (SDA)

This algorithm is due to Kall [8], Kall and Stoyan [9], Frauendorfer and Kall [4], Frauendorfer [5]. See also [11] and [14]. The basic idea is to approximate the original distribution by discrete distributions in a successive manner, via partitions of Ξ, which is an interval covering the support of ξ.

Having the partition $\Xi = \Xi_1 \cup \ldots \cup \Xi_L$, the approximate discrete distribution will be

$$p_k = \mathbf{P}(\xi \mid \xi \in \Xi_k), \;\; \hat{\xi}^k = \mathbf{E}(\xi \mid \xi \in \Xi_k)$$

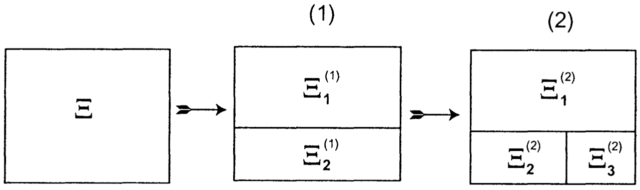

Figure 3. Successive subdivision of Ξ

for $k = 1, \ldots, L$. The key part of the method is the subdivision strategy.

The subdivision strategy is based on lower and upper bounds, for each of the cells in the partition

$$L_k(x^\nu, \xi) \leq \mathrm{E}[\,Q(x^\nu, \xi) \mid \xi \in \Xi_k\,] \leq U_k(x^\nu, \xi),$$

based on the Jensen and on the Edmundson–Madansky inequalities, respectively. For computing the upper bound, the recourse subproblem (8) has to be solved for each of the vertices of Ξ, with ξ taken as the vertex.
$\mapsto$ the cell to be subdivided next will have the maximal relative difference regarding the bounds.
$\mapsto$ the coordinate for the subdivision is selected by employing various heuristic measures of nonlinearity along the corresponding direction.

Great merit of the method: computable error bounds.

4.2 Stochastic decomposition (SD)

This algorithm is due to Higle and Sen [6], [7]. It can be considered as a stochastic, sampling–based version of Benders–decomposition. Let us denote by $\tilde{\xi}^1, \ldots, \tilde{\xi}^\nu, \ldots$ a sample according to the distribution of ξ.

The basic idea is the following: instead of $\mathrm{E}[Q(x,\xi)]$, build Benders–type cuts to the Monte–Carlo approximation

$$\mathrm{E}[Q(x,\xi)] \;\approx\; \frac{1}{\nu} \sum_{k=1}^{\nu} Q(x, \tilde{\xi}^k).$$

This is a moving target, therefore, besides adding new cuts, the existing cuts must also be updated. Sampling and adding cuts runs in a successive manner. The most efficient variants employ “incumbent solutions” and regularized master problems. New cuts are computed by taking into account all previous dual solutions, and the stopping rule is based on bootstrapping.

4.3 Sample average approximation (SAA)

The basic idea of this algorithm has been widely used by practitioners. It became increasing attention due to recent results concerning speed of convergence and judging the quality of the solution, see Shapiro and Homem–de–Mello [24] and Mak, Morton and Wood [17], and the references therein.

The idea is to draw a sample of sample–size L, consider this as a discrete distribution and solve the corresponding two–stage problem. Thus we have

$$f(x) := \mathbf{E}[\,Q(x,\xi)\,] = \frac{1}{L}\,[Q(x,\xi^1) + \ldots + Q(x,\xi^L)]$$

Subsequently the quality of the solution is to be judged, and if needed, the procedure repeated with a larger sample–size.

Crucial issue: judging solution quality. The best estimators are based on the optimality gap between statistical lower and upper bounds.

5. Illustrative computational results

We have randomly generated test problem batteries for two–stage recourse problems, with dimensions A (10×20), W (5×10). T and h are both stochastic and the random vector ξ is 5–dimensional. Each battery consists of 10 test problems.

The batteries were generated as follows: first we have generated a basis–battery with ξ having a normal distribution with stochastically independent components. This has been used to generate 5 further batteries by discretizing the distribution, resulting in test problem batteries with the following amounts of joint ralizations L: $2^{10} = 1'024$, $2^{15} = 32'768$, ; $2^{20} = 1'048'576$, $2^{25} = 33'554'432$, and $2^{30} \approx 1'056'964'608$.

The testing environment was SLP–IOR, our model management system for SLP, developed jointly with P. Kall, see [10], [12].

Computer: 2.6 GHz Pentium-III PC with 1 GB RAM, under the operating system Windows 2000.

Figure 4 displays the minimum, maximum, and average computing times

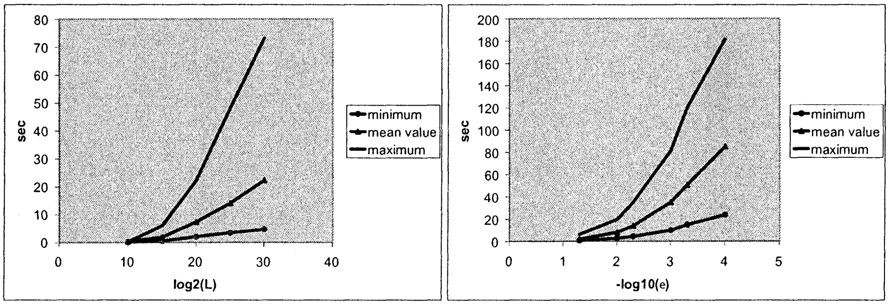

Figure 4. DAPPROX: computing time in seconds

by DAPPROX, our implementation of the successive discrete approximation

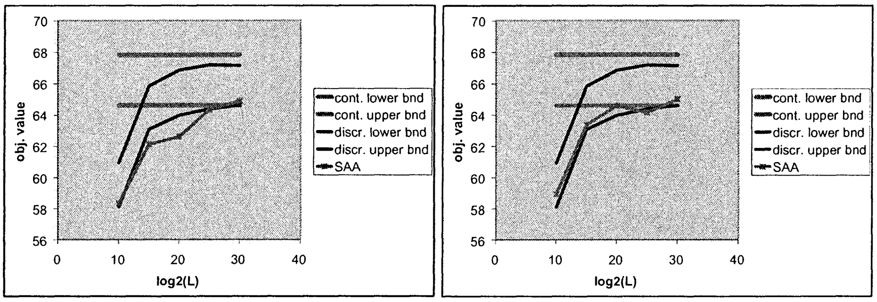

Figure 5. DAPPROX and SAA: objective values at termination

method (jointly developed with P. Kall). On the left–hand–side the dependence of the computing time on L is displayed, whereas the right–hand–side chart shows the dependence on the relative accuracy of the solution. For the latter we took accuracies $\varepsilon = 5 \cdot 10^{-2}$, 10^{-2}, $5 \cdot 10^{-3}$, 10^{-3}, $5 \cdot 10^{-4}$, and 10^{-4}. The computing times were quite acceptable, even for SLP problems with $\sim$ one billion realizations.

In Figure 5 the objective values at termination are displayed, for DAPPROX and SAA, the latter being our implementation of the SAA algorithm. For the computations we took test problem #1. The two horizontal lines correspond to the lower and upper bounds, obtained by DAPPROX for the basis problem with the normal distribution. The approximately parallel increasing curves labeled as "discr. lower bnd" and "discr. upper bnd" correspond to the results obtained by DAPPROX (5% relative accuracy).

For SAA (lowest curve in the left–hand–side chart) we took $L = 500$ and generated 5 samples. After solving the corresponding 5 problems, the objective values of the solutions have been estimated using a sample–size M, which has been chosen for the two charts as $M = 1000$ and $M = 5000$, respectively. The solution with the best estimated objective value was returned by the solver as solution. Observe that the quality of the SAA solution improves dramatically by a relatively moderate change of the run–time parameter M.

References

[1] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4:238–252, 1962.

[2] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer–Verlag, Berlin/Heidelberg, 1997.

[3] J. Elzinga and T.G. Moore. A central cutting plane method for the convex programming problem. *Math. Progr.*, 8:134–145, 1975.

[4] K. Frauendorfer and P. Kall. A solution method for SLP recourse problems with arbitrary multivariate distributions – the independent case. *Probl. Contr. Inf. Theory*, 17:177–205, 1988.

[5] K. Frauendorfer. *Stochastic Two-Stage Programming*. Springer–Verlag, Berlin, 1992.

[6] J.L. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Math. Oper. Res.*, 16:650–669, 1991.

[7] J.L. Higle and S. Sen. *Stochastic decomposition. A statistical method for large scale stochastic linear programming*. Kluwer Academic Publ., 1996.

[8] P. Kall. Approximations to stochastic programs with complete fixed recourse. *Numer. Math.*, 22:333–339, 1974.

[9] P. Kall and D. Stoyan. Solving stochastic programming problems with recourse including error bounds. *Math. Operationsforsch. Statist., Ser. Opt.*, 13:431–447, 1982.

[10] P. Kall and J. Mayer. SLP-IOR: An interactive model management system for stochastic linear programs. *Math. Prog. B*, 75:221–240, 1996.

[11] P. Kall and J. Mayer. *Stochastic linear programming. Models, theory and computation*. Springer-Verlag, 2005.

[12] P. Kall and J. Mayer. Building and solving stochastic linear programming models with SLP-IOR. In S.W. Wallace and W.T. Ziemba, editors, *Applications of Stochastic Programming*, pages 77–90. MPS SIAM, SIAM, Philadelphia, 2005.

[13] P. Kall and J. Mayer. Some insights into the solution algorithms for SLP problems. *Ann. Oper. Res.*, 2005. To appear.

[14] P. Kall and S.W. Wallace. *Stochastic programming*. John Wiley & Sons, Chichester, 1994.

[15] W.K. Klein Haneveld and M.H. van der Vlerk. Integrated chance constraints: reduced forms and an algorithm. *Comp. Management Sci.*, 2005. To appear.

[16] A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Comp. Management Sci.*, 2005. To appear.

[17] W-K. Mak, D.P. Morton, and R.K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.*, 24:47–56, 1999.

[18] K. Marti. *Stochastic optimization*. Springer-Verlag, 2005.

[19] J. Mayer. *Stochastic Linear Programming Algorithms: A Comparison Based on a Model Management System*. Gordon and Breach Science Publishers, 1998.

[20] J. Mayer. On the numerical solution of jointly chance constrained problems. In S. Uryasev, editor, *Probabilistic constrained optimization: Methodology and applications*, pages 220–233. Kluwer Academic Publ., 2000.

[21] Cs. Mészáros. The augmented system variant of IPMs in two–stage stochastic linear programming computation. *Eur. J. Oper. Res.*, 101:317–327, 1997.

[22] A. Prékopa. *Stochastic programming*. Kluwer Academic Publ., 1995.

[23] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Math. Progr.*, 35:309–333, 1986.

[24] A. Shapiro and T. Homem–de–Mello. On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. *SIAM J. Opt.*, 11:70–86, 2000.

[25] R. Van Slyke and R. J-B. Wets. L-shaped linear programs with applications to optimal control and stochastic linear programs. *SIAM J. Appl. Math.*, 17:638–663, 1969.