Comput Optim Appl (2014) 59:617–638
DOI 10.1007/s10589-014-9692-6

A primal-dual aggregation algorithm for minimizing conditional value-at-risk in linear programs

Daniel Espinoza · Eduardo Moreno

Received: 19 August 2013 / Published online: 28 August 2014
© Springer Science+Business Media New York 2014

Abstract Recent years have seen growing interest in coherent risk measures, especially in Conditional Value-at-Risk (CVaR). Since CVaR is a convex function, it is suitable as an objective for optimization problems when we desire to minimize risk. In the case that the underlying distribution has discrete support, this problem can be formulated as a linear programming (LP) problem. Over more general distributions, recent techniques, such as the sample average approximation method, allow to approximate the solution by solving a series of sampled problems, although the latter approach may require a large number of samples when the risk measures concentrate on the tail of the underlying distributions. In this paper we propose an automatic primal-dual aggregation scheme to exactly solve these special structured LPs with a very large number of scenarios. The algorithm aggregates scenarios and constraints in order to solve a smaller problem, which is automatically disaggregated using the information of its dual variables. We compare this algorithm with other common approaches found in related literature, such as an improved formulation of the full problem, cut-generation schemes and other problem-specific approaches available in commercial software. Extensive computational experiments are performed on portfolio and general LP instances.

Keywords Conditional value at risk · Aggregation techniques · Approximation methods · Sample average approximation

D. Espinoza
Department of Industrial Engineering, Universidad de Chile, Santiago, Chile
e-mail: daespino@dii.uchile.cl

E. Moreno (✉)
Faculty of Engineering and Science, Universidad Adolfo Ibañez, Santiago, Chile
e-mail: eduardo.moreno@uai.cl

1 Introduction

Managing to successfully deal with uncertainty and risk has always been a major concern in optimization (see [1–5]), which has many obvious applications. A first mathematical treatment of risk came with the work of Von Neumann and Morgenstern [6]; in which, under some rationality assumptions, the existence of a *risk measure* for agents was assured and linked then agent's attitude towards risk to the concavity of the utility function of each agent. A first algorithmic way to incorporate risk in decision making is probably the seminal work of Markowitz [7], in which the variance of return is used as a proxy of risk. Nonetheless, this variance is introduced as a reasonable way to force diversification in portfolio optimization, rather than as a rigorous way to handle risk in general. An important appeal of the approach was, however, the possibility of actually solving those types of problems. Another important development was the introduction of the *Dual Theory of Choice* of Yaari [8,9], which states that risk aversion and diminishing returns on wealth can be seen as separated aspects, which also introduces a precise notion of *certainty equivalent*. More recently, Artzner et al. [10,11], once again motivated by the portfolio problem, proposed the concept of *Coherent Risk Measures* in the setting of discrete random variables and proposed, as an example, the Conditional Value-at-Risk or CVaR. This was quickly followed up by the work of Rockafellar and Uryasev [12,13], in which the authors present an extended formulation for the problem of evaluating the CVaR, and furthermore, test their ideas on the classical portfolio problem with sampled data but using a CVaR objective function and/or CVaR constraints. The importance of the CVaR was further advanced by the result of Kusuoka [14] which characterized, in a continuous setting and under some extra conditions, all coherent, comonotone, law invariant risk measures (also known as *distortion risk measures* [15]) as combinations of CVaR measures, which makes the CVaR a basis of all such risk measures. Interestingly enough, distortion risk measures are closely related to risk-averse measures introduced early on by Yaari [8].

The extended formulation of the CVaR, by Rockafellar and Uryasev [13], opened up the possibility of using all the approximation machinery from standard stochastic programs (See for example [16] for an in-depth reference, and [17,18] for examples of their use); these results mean that, in theory, under some mild conditions on the domain of the optimization problem and on the characteristics of the space of random variables, solving optimization problems with CVaR objectives can be approximated by solving a series of sampled problems. In particular, for most common portfolio problems, this amounts to solving a series of linear programs (LP). Unfortunately, there are examples [19] in which using too few scenarios leads to strange behavior, prompting the need to solve these problems using large sets of scenarios. This has bolstered research on algorithms and formulations to solve these specially structured LPs. Lim et. al [20] proposed a special-purpose solver for simple portfolio problems with CVaR objective functions. Later on, Ogryczak and Sliwinski [21] compared different formulations for the same problem, thereby showing important advantages of using the dual formulation for these LPs. And more recently, Künzi-bay and Mayer [22] proposed a specialized cut-generation approach (based on the L-shaped method [23]) for the problem. Moreover, similar techniques can be applied to other extensions of the CVaR [24] and other risk measures [25].

In this paper we propose an automatic, exact, primal-dual aggregation method to solve general linear programs (as opposed to the classical portfolio problem) under a CVaR objective function that can deal with a very large number of scenarios. We compare our method with general-purpose solvers and other specialized approaches, under the general set of linear programs found in Netlib [26] as well as on portfolio problems, exploring the issue of scalability and sensibility to the risk-parameter ε of the objective function. These experiments show that the proposed algorithm results in a speed-up factor of 80 when compared with the best general LP-algorithm, and a speed-up factor of 3–20 when compared with other specialized algorithms.

The rest of the paper is organized as follows: Sect. 2 defines the concept of CVaR, introduces linear programs with CVaR objective functions, its representation in the case of finite-support distributions, and the resulting LP problems. Section 3 presents two algorithms to solve the problem found in the related literature. Section 4 describes our algorithm and discusses some extensions to other related problems. Section 5 presents our computational experiments and their results. Finally, in Sect. 6, we summarize our results and deliver our conclusions.

2 Linear programs with CVaR objective functions and discrete distributions

Given $x \in \mathbb{R}^n$, a linear *loss*-function $\hat{z}(x) = -\hat{c}x$, where $\hat{c}$ is a random parameter, and a probability distribution function $F_{\hat{z}}(\lambda) = \mathbb{P}(\hat{z} \leq \lambda)$, the Value-at-Risk (VaR) at level ε is defined as

$$\mathrm{VaR}_\varepsilon(\hat{z}) = \inf\left\{\lambda | F_{\hat{z}}(\lambda) \geq \varepsilon\right\}$$

Similarly, the Conditional Value-at-Risk (CVaR) at level ε can be defined as

$$\mathrm{CVaR}_\varepsilon(\hat{z}) = \min_t \left[t + \frac{1}{\varepsilon}\mathbb{E}((\hat{z} - t)^+)\right],$$

which, assuming that $F_{\hat{z}}(\lambda)$ is continuous at $\lambda = \mathrm{VaR}_\varepsilon(\hat{z})$, is equal to

$$\mathrm{CVaR}_\varepsilon(\hat{z}) = \mathbb{E}\left(\hat{z} | \hat{z} \geq \mathrm{VaR}_\varepsilon(\hat{z})\right).$$

Note that the exact computation of CVaR is not possible except in the case of some particular distribution functions $F_{\hat{z}}$. However, if the underlying distribution is *discrete*, i.e. $\Omega = \{\omega_i\}_{i=1}^N$ and $\mathbb{P}(\hat{\omega} = \omega_i) = p_i$, then, the problem

$$(P) \min \mathrm{CVaR}_\varepsilon(-\hat{c}x) \tag{1a}$$

$$s.t.\ Ax = b \tag{1b}$$

$$x \geq 0, \tag{1c}$$

where $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^m$, $\hat{c} \in \Omega \to \mathbb{R}^n$ is a random variable and $\varepsilon \in]0, 1]$, can be reformulated [13] as

$$(P_N)\ \min t + \frac{1}{\varepsilon}\sum_{i=1}^{N} p_i \eta_i \tag{2a}$$

$$s.t.\ Ax = b \tag{2b}$$

$$c^i x + t + \eta_i \geq 0 \quad \forall i \in \{1, \ldots, N\} \tag{2c}$$

$$x, \eta \geq 0 \tag{2d}$$

Note that Problem (2) requires just one extra variable and one extra constraint for each event in Ω. This, together with the continuous advances in solving linear programs [27], have made this kind of models very attractive. However, as we show in our computational results, there are still many cases in which computing times can be far too long, requiring many hours of computation time.

Note that Problem (2) is exactly the same problem solved when we use a sampled approximation of CVaR for a continuous probability distribution. In this case, c^i is a sampled realization of the random variable $\hat{c}$ and all samples are equiprobable $(p_i = \frac{1}{N})$.

3 Special formulations and special algorithms

These long computation times have led to several articles dealing with how to speed up the problem-solving process of these specially structured LPs. In this section, we present three such alternatives, which we selected because we feel they are representative of the different schemes proposed so far; nonetheless, this is by no means an exhaustive list.

3.1 Improved formulations and general algorithms

A common agreement in the optimization community is that nowadays, if you only want to solve a large LP, the chances are that the best available algorithm will be an interior point algorithm specialized for LPs [27].

However, Problem (2), especially for a large value of N, has a very special structure; and, as was already noted by Ogryczak and Sliwinski [21] in the case of portfolio problems, the dual version of (2) can be written as

$$(D_N)\ \max y^t b \tag{3a}$$

$$s.t.\ y^t A + \frac{1}{\varepsilon}\sum_{i=1}^{N} p_i \lambda_i c^i \leq 0 \tag{3b}$$

$$\sum_{i=1}^{N} p_i \lambda_i = \varepsilon \tag{3c}$$

$$0 \leq \lambda_i \leq 1 \quad \forall i \in \{1, \ldots, N\}. \tag{3d}$$

Note that this problem has only $m+1$ constraints (apart from bounds on variables), and $N+m+1$ variables. This observation is crucial, especially when we consider the "*observed* running time of the simplex algorithm in *real instances*", which according to Dantzig [28], and later on Todd [29], was in the order of $\mathcal{O}(r)$ (where r is the number of rows of the LP). Also more recently this parameter has been considered to be closer to $\mathcal{O}(r\sqrt{c})$ [30] (where c is the number of columns in the problem). This claim is meant to be neither as a precise statement, nor as an exact complexity result, but rather as a practical observation. From this perspective, the results obtained by Ogryczak and Sliwinski [21], are not surprising, but the impact on running time is dramatic. We managed to replicate the results reported by them, in particular the results that indicate that solving the dual problem using primal simplex greatly outperforms all other alternatives, even when compared with interior point algorithms; this last comparison will be reported in our computational results. Given these results, we will use this dual formulation for all problems without trying the primal version.

3.2 Cut-event algorithm

An alternative to solving Problem (1) is to replace the representation of the convex and continuous objective function with its (infinite) sub-differential approximation; more precisely, we can solve problem

$$(O) \min f(x) \tag{4a}$$
$$s.t. \, x \in X, \tag{4b}$$

by solving the problem

$$(O') \min z \tag{5a}$$
$$s.t. \, x \in X \tag{5b}$$
$$z \geq f(x_o) + d(x - x_o) \quad \forall x_o \in X, d \in \partial f(x_o), \tag{5c}$$

where $\partial f(x_o)$ is the subdifferential set of f at x_o. This problem is in general intractable, but in the case of Problem (2), it is equivalent to

$$(\text{CVaR}') \min t + \frac{1}{\varepsilon} w \tag{6a}$$
$$s.t. \, Ax = b \tag{6b}$$
$$w + \sum_{i \in C} p_i (c^i x + t) \geq 0 \quad \forall C \subseteq \{1, \ldots, N\} \tag{6c}$$
$$x \geq 0. \tag{6d}$$

Although (6) still has an exponential number of constraints, it has the flavor of the *sub-tour elimination polytope* (SEP) of the TSP, which can be solved quite efficiently in theory and in practice, even on very large scale problems [31] by using standard cut-generation methods. In the stochastic programming literature, this is known as the

L-shaped method [23], and this idea was proposed by Künzi-Bay and Mayer [22] as a way of minimizing the CVaR. However, taking into account the observations of the previous section, in our implementation we solve the dual problem

$$\text{(CutEv)}\ \max b'y \tag{7a}$$

$$s.t.\ A'y + \sum_{C\subseteq\{1,\dots,N\}} \mu_C \mathbb{E}(\hat{c}|C)\mathbb{P}(C) \le 0 \tag{7b}$$

$$\sum_{C\subseteq\{1,\dots,N\}} \mu_C \mathbb{P}(C) = 1 \tag{7c}$$

$$\sum_{C\subseteq\{1,\dots,N\}} \mu_C = \frac{1}{\varepsilon} \tag{7d}$$

$$\mu \ge 0, \tag{7e}$$

where $\mathbb{P}(C) = \sum_{i\in C} p_i$ and $\mathbb{E}(\hat{c}|C) = \frac{1}{\mathbb{P}(C)} \sum_{i\in C} p_i c^i$. Note that this problem can be solved by column generation, using primal simplex after each re-optimization step. This method is what we call **CutEv** in our computational section.

Another very interesting feature of this approach is that at every stage (i.e. when we solve a *partial* version of (7)) we obtain a lower bound of the original problem. Moreover, taking the candidate (feasible) x^*-solution, by evaluating $\text{CVaR}_\varepsilon(-\hat{c}x^*)$ we obtain a valid upper bound for the problem; thus allowing us to stop our algorithm if the proven gap is small enough. Besides, computing this value is a side effect of finding the most violated *column* to be added to the problem.

3.3 Other specialized algorithms

Given the relevance of portfolio optimization, other specialized algorithms can be found in the literature, and some of them have been made into commercial solvers. For example Lim et al. [20] show a Proximal Bundle Algorithm [32] for this problem, and they also propose an exact three-phase algorithm to solve (1). Nevertheless, the explanation of these algorithms is out of the scope of this paper. We have used the Portfolio Safeguard [33] optimization package for comparison purposes in our computational tests.

4 Primal-dual aggregation method (PDAgg)

4.1 Description of the algorithm

The central idea of our algorithms follow from the following observation: under the assumption of $N \ggg m$, almost all λ variables in Problem (3), will be at one of the bounds in any (extreme) optimal solution; i.e., they will have the same value. Moreover, in our experiments, the number of λ variables that are basic is very small, usually up to three. Thus, if we could *guess* the different values these variables will assume in the optimal solution, $\{\lambda_k^*\}_{k=1}^K$, then, we could rewrite (3) using only $n + 1$

constraints (apart from bounds), and $m + K$ variables. Our algorithm starts out by assuming that all λ variables have the same value, obtains a candidate x^*, evaluates $\text{CVaR}_\varepsilon(-\hat{c}x^*)$, and then identifies the set of λ variables that support this value, and allow the approximated problem to support the same set of values, and iterate the process. Interestingly, this idea of forcing a set of λ variables to have the same value in the dual exactly corresponds in the primal to aggregate those set of scenarios into a single scenario. This idea of forcing extra constraints (in our case, fixing many λ variables at the same value) into a large problem to solve it more efficiently was proposed, in a different context, by Bienstock and Zuckerberg [34].

More precisely, let us consider $\mathcal{N} = \{N_k\}_{k=1}^K$ a partition of $\{1, \dots, N\}$, and the problem

$$(\underline{D}_{\mathcal{N}})\ \max y^t b \tag{8a}$$

$$s.t.\ y^t A + \frac{1}{\varepsilon}\sum_{k=1}^{K} \bar{p}_k \lambda_k \bar{c}^k \le 0 \tag{8b}$$

$$\sum_{k=1}^{K} \bar{p}_k \lambda_k = \varepsilon \tag{8c}$$

$$0 \le \lambda_k \le 1 \quad \forall k \in \{1, \dots, K\}, \tag{8d}$$

where $\bar{p}_j = \mathbb{P}(N_k) = \sum_{i \in N_k} p_i$ and where $\bar{c}^k = \mathbb{E}(\hat{c}|N_k) = \sum_{i \in N_k} p_i \hat{c}^i$. Note that this problem can be seen as an *aggregated* version of (3) according to partition $\mathcal{N}$. In fact, it is easy to see that for any partition $\mathcal{N}$ of $\{1, \dots, N\}$, (8) is obtained from (3) by adding the constraints $\lambda_i = \lambda_j, \forall i, j : \exists N_k \in \mathcal{N}, i, j \in N_k$. Hence, $z_{\underline{D}_{\mathcal{N}}}$ is a lower bound of (3) for any partition $\mathcal{N}$ whatsoever.

On the other hand, given $\tilde{x} \in \mathbb{R}^n$, satisfying $A\tilde{x} = b,\ \tilde{x} \ge 0$, problem

$$(\overline{D}_{\tilde{x}})\ \max\ -\frac{1}{\varepsilon}\sum_{i=1}^{N} p_i \lambda_i c^i \tilde{x} \tag{9a}$$

$$s.t. \sum_{i=1}^{N} p_i \lambda_i = \varepsilon \tag{9b}$$

$$0 \le \lambda_i \le 1 \quad \forall i = 1, \dots, N \tag{9c}$$

corresponds to the dual of (2) after fixing $x = \tilde{x}$. Hence, $z_{\overline{D}_{\tilde{x}}}$ is an upper bound for (2), and also for (3).

In summary, we have that

$$z_{\underline{D}_{\mathcal{N}}} \le z_{D_N} \le z_{\overline{D}_{\tilde{x}}}$$

for any partition $\mathcal{N}$ of $\{1, \dots, N\}$ and for any $\tilde{x}$ such that $A\tilde{x} = b, \tilde{x} \ge 0$.

Note that (9) is simply a continuous knapsack problem, whose solution can be computed in $\mathcal{O}(N \log(N))$ time using a greedy algorithm [35]. Moreover, any extreme optimal solution of (9) has at most three different values for λ_i: 0, 1 and a fractional value (if required) to satisfy (9b). These values induce a partition of $\{1,\ldots,N\} = \{i : \lambda_i = 0\} \bigcup \{i : \lambda_i = 1\} \bigcup \{i : \lambda_i \in (0,1)\}$, that we denote $\mathcal{N}^*_{\tilde{x}}$.

Interestingly enough, if we take $\tilde{x}$ as a dual optimal solution for (8b), and we have that $\mathcal{N}^*_{\tilde{x}}$ is *representable* in $\mathcal{N}$ (i.e., $\mathcal{N}^*_{\tilde{x}}$ can be obtained by unions of elements of $\mathcal{N}$), then $z_{\underline{D}_{\mathcal{N}}} = z_{\overline{D}_{\tilde{x}}}$, which means that we have solved (3). This is the key idea that motivates our algorithm, presented in Algorithm 1.

Algorithm 1 PDAgg(δ)

```
Require: Stopping gap $\delta \geq 0$
1: $\mathcal{N} \leftarrow \{\{1,\ldots,N\}\}$
2: loop
3:   Solve ($\underline{D}_{\mathcal{N}}$). Let $\tilde{x}$ be a dual-optimal solution of (8b) and $z_{\underline{D}_{\mathcal{N}}}$ its objective value.
4:   Solve ($\overline{D}_{\tilde{x}}$). Let $\mathcal{N}^*_{\tilde{x}}$ be an induced partition of ($\overline{D}_{\tilde{x}}$), and $z_{\overline{D}_{\tilde{x}}}$ its optimal objective value.
5:   if $z_{\underline{D}_{\mathcal{N}}} - z_{\overline{D}_{\tilde{x}}} \leq \delta$ then
6:     return Solution ($\tilde{x}$)
7:   else
8:     Let $\mathcal{N} \leftarrow \{N_i \cap N_j : N_i \in \mathcal{N},\ N_j \in \mathcal{N}^*_{\tilde{x}}\}$.
```

Lemma 1 *Algorithm 1 for* $\delta = 0$ *stops and returns the optimal solution of* (2)

Proof We only need to prove that at every step the number of elements in the partition $\mathcal{N}$ is strictly increasing, thus bounding the number of iterations of the overall algorithm. Let us denote $L(\underline{D}_{\mathcal{N}}, \mu)$ the Lagrangian relaxation of $\underline{D}_{\mathcal{N}}$ obtained by penalizing constraints (8b) by μ. That is

$$\begin{aligned} L(\underline{D}_{\mathcal{N}}, \mu) \quad := \quad & \max\ y^t b + \mu\left(-y^t A - \tfrac{1}{\varepsilon} \sum_{k=1}^{K} \bar{p}_k \lambda_k \bar{c}^k\right) \\ & s.t. \sum_{k=1}^{K} \bar{p}_k \lambda_k = \varepsilon \\ & \quad\quad 0 \leq \lambda_k \leq 1 \quad \forall k = 1,\ldots,K. \end{aligned}$$

Let us suppose that the size of $\mathcal{N}$ does not increase after step 8, and let λ^* be the optimal solution of ($\overline{D}_{\tilde{x}}$). That is, the $\mathcal{N}^*_{\tilde{x}}$ obtained in step 4 is a coarser partition of $\{1,\ldots,N\}$ than $\mathcal{N}$. Since the values of λ^* are constant within each set of the partition given by $\mathcal{N}^*_{\tilde{x}}$, it follows that $\lambda^*_i = \lambda^*_{i'}$ for all $i, i' \in N_k, \forall k = 1\ldots K$. Therefore, we can define $\hat{\lambda}^*_k = \lambda^*_i$ for some $i \in N_k$, and $\hat{\lambda}^*$ is feasible for $L(\underline{D}_{\mathcal{N}}, \mu)$ for any $\mu \geq 0$, in particular for $\mu = \tilde{x}$. However, since $\tilde{x}$ is the optimal dual of constraints (8b), then $\hat{\lambda}^*$ is optimal for $L(\underline{D}_{\mathcal{N}}, \tilde{x})$, and its value is equal to $z_{\underline{D}_{\mathcal{N}}}$ and $z_{\overline{D}_{\tilde{x}}}$, thus proving the result. □

The efficiency of the proposed algorithm is based on three facts: Firstly, the aggregated problem $\underline{D}_{\mathcal{N}}$ can be much smaller than the original problem. Secondly, the optimal solution of $\overline{D}_{\tilde{x}}$ has a few different values, resulting in a coarse refinement of

$\mathcal{N}$, which induce very small instances of $\underline{D}_{\mathcal{N}}$. And finally that problem $\overline{D}_{\tilde{x}}$ can be efficiently solved, even for a large N.

Note that, unlike most of the classical aggregation techniques for stochastic two-stage problems [36,37], where the focus is to avoid the resolution of a subproblem over all scenarios, we do perform a sub-problem optimization step looking at all scenarios. Although in theory we could end up with a completely refined partition (i.e. $\mathcal{N} = \{1, \dots, N\}$ and then solving (8) will be equivalent to solve the original problem (3)), we will see in the computational results, that only a few iterations are required to solve the problem, which means that for most instances the resulting final partition $\mathcal{N}^*$ satisfies that $|\mathcal{N}^*| \lll N$.

4.2 Extension to other objective functions

In the previous section we described a new primal/dual method for handling general linear programs with a CVaR objective function. However, the proposed technique can also be adapted to other risk measures.

4.2.1 Weighted CVaR

A first natural extension is to consider objective functions that are positive combinations of CVaR at different probability levels, i.e. to consider objective functions of the form

$$f(x) = \sum_{r=1}^{R} \gamma_r \mathrm{CVaR}_{\varepsilon_r}(-\hat{c}x),$$

for $R \in \mathbb{N}$, $\gamma \in \mathbb{R}^R_+$. Note that, by scaling arguments, it is enough to consider γ that satisfies $\sum_{r=1}^{R} \gamma_r = 1$; and consequently, $f(x)$ is just another coherent risk measure for $\hat{c}x$. In this case, given samples $\{c^{ir}\}_{i=1}^{N^r}$ for $r = 1, \dots, R$, the problem we would like to solve can be written as

$$(P_N^R) \min \sum_{r=1}^{R} \gamma_r \left(t_r + \frac{1}{\varepsilon_r} \sum_{i=1}^{N^r} p_{ir} \eta_{ir} \right) \tag{10a}$$

$$s.t.\ Ax = b \tag{10b}$$

$$c^{ir}x + t_r + \eta_{ir} \geq 0 \quad \forall r \in \{1, \dots, R\},\ i \in \{1, \dots, N^r\} \tag{10c}$$

$$x, \eta \geq 0, \tag{10d}$$

and its corresponding dual can be written as

$$(D_N^R) \max y^t b \tag{11a}$$

$$s.t.\ y^t A + \sum_{r=1}^{R} \left(\frac{\gamma_r}{\varepsilon_r} \sum_{i=1}^{N^r} p_{ir} \lambda_{ir} c^{ir} \right) \leq 0 \tag{11b}$$

$$\sum_{i=1}^{N^r} p_{ir}\lambda_{ir} = \varepsilon_r \quad \forall r \in \{1,\dots,R\} \tag{11c}$$

$$0 \le \lambda_{ir} \le 1 \quad \forall r \in \{1,\dots,R\},\ i \in \{1\dots N^r\}. \tag{11d}$$

Note that this problem has almost the same structure as (3); and given a set of partitions $\{\mathcal{N}^r\}_{r=1}^R$, where $\mathcal{N}^r$ is a partition of $\{1,\dots,N^r\}$, we can use the same upper/lower bounds and method defined in Sect. 4; however, its complexity will increase linearly in the number of components of the objective function.

4.2.2 Minimax objective function

Another classical objective function for portfolio optimization [38] is the so-called minimax portfolio selection rule, which can be seen as a worst case optimization approach. In this case, given a sample of objective coefficients $\{c^i\}_{i=1}^R$, the minimax objective function can be written as

$$(P_{\text{minimax}})\ \min t \tag{12a}$$

$$s.t.\ Ax = b \tag{12b}$$

$$t + c^i x \ge 0 \quad \forall i \in \{1,\dots,N\} \tag{12c}$$

$$x \ge 0, \tag{12d}$$

and its dual can be written as

$$(D_{\text{minimax}})\ \max y^t b \tag{13a}$$

$$s.t.\ y^t A + \frac{1}{\varepsilon}\sum_{i=1}^{N} p_i\lambda_i c^i \le 0 \tag{13b}$$

$$\sum_{i=1}^{N} p_i\lambda_i = \varepsilon \tag{13c}$$

$$0 \le \lambda_i \le 1 \quad \forall i \in \{1,\dots,N\}, \tag{13d}$$

where $\varepsilon = \frac{1}{N}$ and $p_i = \frac{1}{N}$ for $i \in \{1,\dots,N\}$. Note that (13) has exactly the same form as (3), which is why we can directly use our proposed algorithm.

*4.2.3 Mean-absolute deviation (*MAD*) objective functions*

The Mean-Absolute Deviation was introduced by Konno and Yamazaki [39] in the context of portfolio optimization and can be seen as an L_1 approximation of the L_2 model by Markowitz [7]. The MAD objective function is $\mathbb{E}(|\hat{c}x - \mathbb{E}(\hat{c}x)|)$. If we assume a sample $\{c^i\}_{i=1}^N$ with probabilities p_i of the objective coefficients, the related optimization problem can be stated as

$$
\begin{aligned}
(P_{\text{MAD}})\ \min\ & \sum_{i=1} p_i \left(\eta_i^+ + \eta_i^-\right) && \text{(14a)}\\
s.t.\ & Ax = b && \text{(14b)}\\
& \eta_i^+ - \eta_i^- + c^i x - t = 0 \quad \forall i \in \{1, \ldots, N\} && \text{(14c)}\\
& t - \bar{c}x = 0 && \text{(14d)}\\
& x, \eta^+, \eta^- \geq 0, && \text{(14e)}
\end{aligned}
$$

where $\bar{c} := \sum_{i=1}^N p_i c^i$. Its dual can be written as

$$
\begin{aligned}
(D_{\text{MAD}})\ \max\ & y^t b && \text{(15a)}\\
s.t.\ & y^t A - \lambda_o \bar{c} + \sum_{i=1}^N p_i c^i \leq 0 && \text{(15b)}\\
& \sum_{i=1}^N p_i \lambda_i = \lambda_o && \text{(15c)}\\
& -1 \leq \lambda_i \leq 1 \quad \forall i \in \{1, \ldots, N\}. && \text{(15d)}
\end{aligned}
$$

Unfortunately, Problem (15) does not have the same form of Problem (3), but we can use a similar trick to get upper and lower bounds for an aggregated problem. More precisely, given $\mathcal{N} = \{N_k\}_{k=1}^K$ a partition of $\{1, \ldots, N\}$, $\tilde{x}$ satisfying $A\tilde{x} = b$, $\tilde{x} \geq 0$, and substituting λ_o, we can define

$$
\begin{aligned}
(\underline{D}_{\text{MAD},\mathcal{N}})\ \max\ & y^t b && \text{(16a)}\\
s.t.\ & y^t A + \sum_{k=1}^K \bar{p}_k \lambda_k (\bar{c}^k - \bar{c}) \leq 0 && \text{(16b)}\\
& -1 \leq \lambda_k \leq 1 \quad \forall k \in \{1, \ldots, K\}, && \text{(16c)}
\end{aligned}
$$

and

$$
\begin{aligned}
(\overline{D}_{\text{MAD},\tilde{x}})\ \max\ & \sum_{i=1}^N \lambda_i p_i \left(\bar{c} - c^i\right) \tilde{x} && \text{(17a)}\\
s.t.\ & -1 \leq \lambda_i \leq 1 \quad \forall i \in \{1, \ldots, N\}. && \text{(17b)}
\end{aligned}
$$

Using the same arguments as in Sect. 4, it is easy to prove that Problem (16) is a lower bound of Problem (15), and that Problem (17) is an upper bound of Problem (15). Furthermore, by its structure, any optimal solution to (17) can have at most two different values for λ_i, $i \in \{1, \ldots, N\}$; hence, the same refinement arguments apply, while the complexity of solving Problem (17) is just $\mathcal{O}(N)$.

5 Computational results

5.1 Definitions and settings

In order to benchmark the performance of the proposed methodology, we implemented the following algorithms and techniques to solve Problem (2):

CPX: We formulated Problem (2) and solved it using the CPLEX Barrier algorithm.
CPX-Dual: As suggested in [21], we formulated the dual of the problem, detailed in (3) and solved this dual formulation using the Primal Simplex algorithm.
CutEv: *Cut-event* formulation, presented in (7) and solved using column generation.
PDAgg: Our proposed methodology, presented in Algorithm 1.

We compared previous methodologies using two set of instances: general LP problems and a set of large portfolio problems. For general LP problems, we used the Netlib [26] LP problem collection. In order to introduce uncertainty into the objective function, we modified the original objective function by multiplying independently each coefficient of the objective function by a random variable χ. We tested two random variables:

- $\chi \sim U[0, 1]$, i.e. χ is a uniform distribution in [0,1].
- $\chi \sim \begin{cases} \mathcal{N}(1, 0.4) & \text{with probability } 0.95 \\ \exp(10) & \text{otherwise} \end{cases}$, i.e. in 95% of the cases, χ follows a normal distribution; otherwise, it follows an exponential distribution. This is one of the distributions used in [19].

Furthermore, we tested different risk levels for the CVaR, namely $\varepsilon \in \{0.001, 0.01, 0.05, 0.10, 0.25, 0.50, 0.75, 0.90, 0.99 \text{ and } 0.999\}$, and ten different sample sizes $|\mathcal{N}|$ from 10.000 to 100.000 samples in increments of 10.000. This gives us a total of 200 configurations for each Netlib instance.

All our code was implemented using the C programming language. All runs where made using a single thread with an address space limit and data limit of 2 Gb and a 20-h running time limit. The machines were running Linux 2.6.18 under x86_64 architecture, with two quad-core Intel® Xeon® E5620 processors and with 48 Gb of RAM. To obtain better time measurements, the machines were configured (in BIOS) with the following technologies disabled: Intel® Turbo Boost Technology, Intel® Hyper-Threading Technology, and Intel® Virtualization Technology (VT-x). These settings allowed us to run up to eight instances in a machine without (much) interference between processes.

A final detail is the termination condition of the algorithm. Since both **CutEv** and **PDAgg** provide an upper and lower bound for the problem at each iteration, we stopped whenever the relative gap between these bounds was less than 10^{-6}. The value 10^{-6} was chosen because it is the default reduced cost tolerance for the CPLEX simplex algorithm, which should make all approaches comparable.

Table 1 Comparison of **PDAgg** and **CutEv** on Netlib instances under different stopping criteria

	# Inst.	**PDAgg**			**CutEv**		
		10^{-2}	10^{-4}	10^{-6}	10^{-2}	10^{-4}	10^{-6}
Global	16,600	0.392	0.776	1.017	0.459	1.507	2.714
>1 s	9,541	1.360	3.663	5.293	1.555	6.949	14.900
>10 s	5,411	2.828	10.365	15.710	2.946	19.276	48.111
>100 s	2,303	5.013	30.894	50.716	4.027	51.408	160.112
>1,000 s	573	9.471	114.207	194.492	5.025	176.010	680.784

5.2 Overall comparisons

Table 3 (in Appendix) shows the geometric mean [40] running time between the 200 configurations for each instance. In this table, we only include the 56 instances that can be solved by the four algorithms within our prescribed limits.

It can be seen that formulating the full problem and solving it using the CPLEX barrier algorithm (Column **CPX** in the table) is the slowest, with an average running time of 29.957 s. Also, we can see that the observed behavior of **CPX-dual** for portfolio problems [21] also applies to general CVaR minimization problems, with an average running time of 8.279 s. Additionally, it can be seen that our algorithm (Column **PDAgg**) and the *cut-event* formulation (Column **CutEv**) are much faster than formulating the full problem, with an average of 0.373 and 1.127 s respectively.

5.3 Comparing **PDAgg** and **CutEv** algorithms

Since these last two algorithms are much faster than both algorithms for the monolithic formulation, we will focus on the results of **PDAgg** and **CutEv**. In this case, a total of 83 Netlib's instances were solved by both algorithms within our prescribed limits. We only excluded the instances `80bau3b, fit2p, fit2d, ship12l` and `ship12`, because some configuration of these problems exceed the memory limit of 2Gb.

5.3.1 The effect of the stopping criteria

In Table 1 we show the geometric mean running time of both methods over the 16,600 problems solved by each algorithm, under different stopping criteria (namely, a relative gap less than 10^{-2}, 10^{-4} and 10^{-6}). In order to improve the accuracy of the results, we also show these averages over subsets of problems in which at least one of the algorithms requires more than 1, 10, 100 and 1,000 s to achieve the final gap of 10^{-6}.

As we can see, both methods require a similar time to solve instances with a gap of 10^{-2} with a slight advantage for the **CutEv** algorithm, especially on the hardest problems. However, our algorithm outperforms the *cut-event* approach when we require more precision.

5.3.2 The effect of the sampling size and the risk parameter

Figure 1 shows the geometric mean running time of both algorithms on the instances grouped by different levels of ε and by different sample sizes. It can be seen that our algorithm outperforms the *cut-events* approach, especially when the number of samples increases; moreover, it is clear that the growth rate of the running time with respect to N for **PDAgg** is smaller than for **CutEv**, which is an indicator of a better scalability (in terms of size) of our algorithm. Something interesting can be seen when we study the effect of ε on the running time. Both algorithms are slower for small values of ε, and faster for large values of ε. This was somewhat unexpected, and this may be because for larger ε values, we are closer to optimizing the plain expected value. But again, it must be noted that our algorithm greatly outperforms **CutEv** especially for small values of ε (which is incidentally the most common case), especially in problems taken from the risk-averse optimization literature.

5.3.3 Portfolio problems

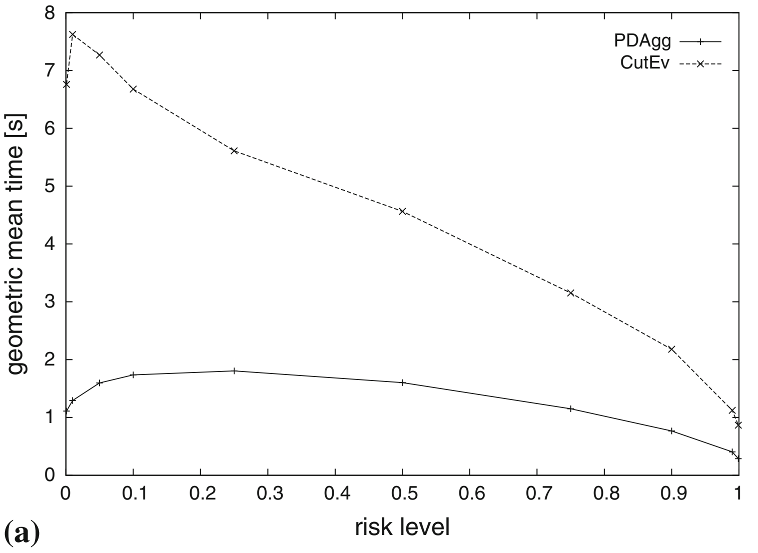
The purpose of this second set of instances is to compare our algorithm with other specialized algorithms for portfolio optimization. A large portfolio problem was constructed following [41]: we used the monthly closing price of 4,553 stocks from January 2008 to December 2012, obtained using CRSP Monthly Stock data from Wharton Research Data Service. From this set, we randomly selected a sample of 500 stocks and we computed the expected returns and the covariance matrix using the methodology described in [42]. Using these data, we generated $100,000$ scenarios from a multivariate normal distribution with the corresponding parameters.

For this experiment, we additionally benchmarked our approach against the algorithm of [20], implemented in AORDA Portfolio Safeguard [33]. The experiments with this last algorithm was run in a similar architecture but in a Microsoft Windows environment, and with the same stopping criteria (precision of 10^{-6}).

Table 2 shows the computation times of these instances. The results are similar to those obtained using Netlib's instances. For this particular problem, the *cut-event* formulation and the specialized algorithm are faster than our algorithm for ε equal to 0.25, 0.5 and 0.75. However, our algorithm outperforms the other two algorithms for more extreme values of ε. This is particularly interesting because for portfolio problems, the commonly desired value of ε is near to zero, while our algorithm is more than 10 times faster than the other algorithms.

5.3.4 Detailed results for **PDAgg**

Table 4 (in Appendix) shows the running details of our algorithms on the portfolio problems and on the 83 Netlib instances using 100,000 scenarios. We present the geometric mean running time (in seconds) required to solve the problem, the size of the final partition (column $|\mathcal{N}|$) and the average number of iterations required (column Iter), among the different distributions and risk-levels. We also include the size of each original problem, in terms of the number of variables, constraints, and the number of non-zero coefficients of the objective function, which represent the dimension of the



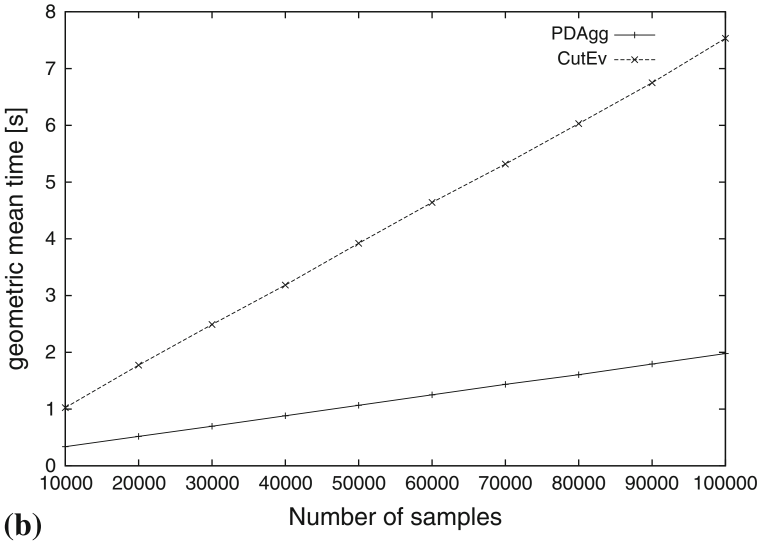


Fig. 1 Geometric mean running times of **PDAgg** and **CutEv** on Netlib instances. **a**Time versus ε **b** Time versus N

sample space. It can be seen that the efficiency of our algorithm is partially accounted for by the small number of iterations required, which results in a partition of small size, hence each sub-problem can be solved quickly. In fact, instead of solving a problem with 100,000 scenarios, in the end our algorithm had a geometric average of 5.97 iterations using only 43.87 aggregated scenarios. It is worth noting that for several problems, particularly the problems with a low-dimensional sample space, the

Table 2 Computation times on portfolio instances

ε	**PDAgg**	**CutEv**	**AORDA**
0.001	5.92	85.07	54.51
0.01	7.93	99.22	98.48
0.05	24.09	103.56	113.39
0.10	117.14	102.28	125.91
0.25	561.91	106.50	144.32
0.50	584.92	136.98	177.12
0.75	465.65	127.21	107.55
0.90	28.69	47.07	41.91
0.99	2.60	7.09	9.11
0.999	0.43	2.28	1.84

algorithm requires only two iterations to solve problem, obtaining the optimal partition of the scenarios in a few seconds.

6 Conclusions

We have shown a primal-dual aggregation technique to exactly solve CVaR minimization problems under discrete probability distributions, with a very large number of scenarios. This framework also comprises the sampled approximation of CVaR minimization problems under arbitrary probability distributions. Computational experiments show that this method is much more efficient in terms of running time than other general and specialized algorithms for this problem. Moreover, our algorithm showed a better scalability on the number of scenarios than the competing algorithms, and has the advantage of handling general LP constraints. This might be due because the optimal solution to these problems usually requires a far smaller number of *representative* scenarios to prove optimality. An interesting open question is how to find a better selection of these representative scenarios, which might allow us to obtain even smaller aggregations, in order to further improve the performance of this algorithm.

We would also like to note that this primal-dual aggregation technique could also be applied to solve some other special-structured two-stage stochastic programming problems, opening a new direction for future research. Finally, the proposed framework (as well as the cut-event approach) can be further expanded in order to deal with problems having several CVaR constraints and/or objective functions, thereby further extending its applicability.

Acknowledgments Daniel Espinoza was partially funded by the FONDECYT Grant 1110024 and Millenium Nucleus Information and Coordination in Networks ICM/FIC P10-024F. Eduardo Moreno acknowledges the financial support of the FONDECYT Grant 1130681 and Basal Center CMM-UCh. Both authors acknowledge Stan Uryasev for providing AORDA software for benchmarking purposes.

Appendix: Detailed tables of results

See Tables 3 and 4.

Table 3 Results of all algorithms on running Netlib instances

Name	**PDAgg**	**CPX-Dual**	**CutEv**	**CPX**
25fv47	2.979	227.412	7.897	2823.874
adlittle	1.773	45.729	9.471	66.804
afiro	0.045	0.413	0.162	1.364
agg2	2.076	116.208	31.119	295.196
agg3	1.073	88.085	11.633	303.347
bandm	0.996	33.523	5.044	127.979
beaconfd	0.212	6.993	0.401	45.023
boeing1	3.088	82.197	55.529	741.207
boeing2	0.769	19.458	5.040	108.206
bore3d	0.080	9.219	0.283	21.642
brandy	0.058	0.377	0.161	1.775
capri	0.371	2.358	1.436	6.860
degen2	3.916	668.789	19.976	882.728
e226	1.837	62.300	18.394	187.984
etamacro	0.242	11.452	0.431	44.608
fffff800	0.250	1.921	0.427	3.235
finnis	1.042	69.745	2.186	1040.445
ganges	0.557	8.265	3.468	76.688
greenbea	10.656	89.550	6.473	1203.021
greenbeb	4.576	127.795	4.441	1046.617
grow15	0.188	3.632	0.542	22.783
grow22	0.442	5.593	0.922	41.402
grow7	0.171	1.347	0.406	6.174
israel	0.380	19.627	1.265	62.674
kb2	0.051	0.382	0.145	1.479
lotfi	0.110	2.053	0.231	2.758
nesm	2.366	190.897	2.936	5050.060
perold	0.452	3.215	0.582	3.774
pilot	8.530	39.183	15.557	30.690
pilot.ja	0.848	6.457	1.124	12.999
pilot.we	18.668	316.110	64.417	227.635
pilot4	0.205	1.952	0.325	2.855
recipe	0.068	3.260	0.256	17.871

Table 3 continued

Name	**PDAgg**	**CPX-Dual**	**CutEv**	**CPX**
sc105	0.036	0.227	0.136	0.872
sc205	0.039	0.233	0.141	0.915
sc50a	0.034	0.175	0.134	0.745
sc50b	0.035	0.174	0.134	0.846
scagr25	0.614	159.899	1.712	640.479
scagr7	0.179	24.360	0.458	55.400
scfxm1	0.078	3.024	0.197	4.568
scfxm2	0.119	5.901	0.270	12.914
scfxm3	0.172	9.188	0.364	25.703
scorpion	1.130	54.413	3.756	392.443
scsd1	9.582	225.561	47.585	3432.556
sctap1	5.742	386.091	110.350	865.274
seba	1.088	46.020	3.502	411.507
share1b	0.130	2.587	0.284	14.688
share2b	0.188	3.500	0.346	19.146
stair	0.112	0.731	0.267	1.365
standata	0.066	0.985	0.181	3.114
standgub	0.066	1.030	0.184	3.074
standmps	0.057	1.001	0.183	4.692
stocfor1	0.164	3.318	0.401	11.764
tuff	0.058	1.080	0.181	1.861
vtp.base	0.058	1.078	0.163	2.418
wood1p	0.240	5.645	0.483	9.805
woodw	0.627	5.983	0.856	14.044
Total	0.373	8.279	1.127	29.957

Table 4 Results of **PDAgg** algorithm on Netlib and portfolio instances with 100,000 scenarios

Name	Ncols	Nrows	Obj	Time	# Sets	# Iter
25fv47	1571	821	727	4.61	271.35	9.70
adlittle	97	56	82	4.03	3445.20	15.60
afiro	32	27	5	0.11	4.60	2.90
agg	163	488	131	1.18	598.60	11.45
agg2	302	516	231	5.24	1566.20	12.80
agg3	302	516	231	2.43	727.30	11.15
bandm	472	305	165	2.18	1013.45	12.95
beaconfd	262	173	101	0.46	25.00	5.90
bnl1	1175	643	1008	8.11	739.40	11.10

Table 4 continued

Name	Ncols	Nrows	Obj	Time	# Sets	# Iter
bnl2	3489	2324	2125	25.55	930.80	12.45
boeing1	384	351	380	6.58	1840.40	13.45
boeing2	143	166	143	1.84	1124.25	13.00
bore3d	315	233	96	0.17	2.00	2.00
brandy	249	220	2	0.12	3.00	2.50
capri	353	271	19	0.77	1592.75	11.95
cycle	2857	1903	602	38.64	6358.50	18.90
czprob	3523	929	3504	539.68	3600.20	15.30
d2q06c	5167	2171	3257	381.67	4659.35	16.55
d6cube	6184	415	6184	215.99	1133.45	13.15
degen2	534	444	471	8.23	1298.25	12.40
degen3	1818	1503	1584	83.53	2448.45	13.65
e226	282	223	189	4.11	2374.20	14.25
etamacro	688	400	80	0.44	18.05	5.15
fffff800	854	524	8	0.45	78.15	8.30
finnis	614	497	404	2.13	163.90	9.95
fit1d	1026	24	1026	2.78	52.40	6.40
fit1p	1677	627	1026	1.77	9.00	4.00
ganges	1681	1309	109	1.16	538.10	10.10
greenbea	5405	2392	622	11.62	57.65	6.15
greenbeb	5405	2392	622	5.35	17.65	4.80
grow15	645	300	45	0.29	74.75	4.50
grow22	946	440	66	0.66	53.95	6.70
grow7	301	140	21	0.38	90.05	7.70
israel	142	174	89	0.82	324.50	10.65
kb2	41	43	5	0.12	3.80	2.80
lotfi	308	153	8	0.28	21.20	7.05
maros	1443	846	392	8.99	1852.15	14.35
maros-r7	1443	846	392	18.07	72.30	5.80
modszk1	1620	687	990	4.87	411.80	11.70
nesm	2923	662	700	3.16	20.85	5.10
perold	1376	625	8	0.50	2.70	2.35
pilot	3652	1441	53	9.28	967.45	11.90
pilot.ja	3652	1441	53	0.94	3.20	2.60
pilot.we	2789	722	92	30.72	6773.30	18.00
pilot4	1000	410	4	0.28	5.85	3.70
pilot87	4883	2030	652	43.59	798.20	11.65
pilotnov	2172	975	72	3.93	1974.05	13.50

Table 4 continued

Name	Ncols	Nrows	Obj	Time	# Sets	# Iter
portfolio	500	1	500	30.29	7596.30	17.80
recipe	180	91	89	0.16	2.00	2.00
sc105	103	105	1	0.09	2.00	2.00
sc205	203	205	1	0.09	2.00	2.00
sc50a	48	50	1	0.08	2.00	2.00
sc50b	48	50	1	0.09	2.00	2.00
scagr25	500	471	475	1.26	73.50	6.50
scagr7	140	129	133	0.42	17.65	4.85
scfxm1	457	330	23	0.16	3.50	2.75
scfxm2	914	660	46	0.21	3.60	2.80
scfxm3	1371	990	69	0.28	4.05	2.90
scorpion	358	388	282	2.33	782.95	12.65
scrs8	1169	490	847	6.29	1209.60	11.10
scsd1	760	77	760	26.98	2800.65	15.00
scsd6	1350	147	1350	84.83	3429.45	15.10
scsd8	2750	397	2750	72.07	1609.50	13.05
sctap1	480	300	360	15.91	2284.55	13.35
sctap2	1880	1090	1410	37.58	1534.25	13.05
sctap3	2480	1480	1860	55.98	1805.80	13.20
seba	1028	515	522	2.43	268.60	9.80
share1b	225	117	31	0.29	15.40	6.15
share2b	79	96	36	0.43	22.40	7.65
shell	1775	536	1344	6.12	97.05	9.90
ship04l	2118	402	2118	8.82	146.10	9.45
ship04s	1458	402	1458	5.76	140.55	9.30
ship08l	4283	778	4283	27.14	584.95	11.10
ship08s	2387	778	2387	16.21	518.90	10.60
stair	467	356	1	0.16	2.00	2.00
standata	1075	359	7	0.14	3.70	2.85
standgub	1184	361	7	0.14	3.70	2.85
standmps	1075	467	7	0.11	2.00	2.00
stocfor1	111	117	27	0.39	51.40	7.90
stocfor2	2031	2157	1149	2.60	19.50	4.65
tuff	587	333	3	0.11	2.00	2.00
vtp.base	203	198	6	0.13	3.30	2.65
wood1p	2594	244	1	0.28	2.00	2.00
woodw	8405	1098	4	0.68	9.60	4.05

References

1. Avriel, M., Williams, A.: The value of information and stochastic programming. Oper. Res. **18**(5), 947–954 (1970)
2. Dantzig, G.B.: The diet problem. Interfaces **20**(4), 43–47 (1990)
3. Stigler, G.J.: The cost of subsistence. J. Farm Econ. **27**(2), 303–314 (1945)
4. Tintner, G.: Stochastic linear programming with applications to agricultural economics. In: Proceedings of the Second Symposium in Linear Programming, vol. 1, pp. 197–228. National Bureau of Standards Washington, DC (1955)
5. Wets, R.J.B.: Programming under uncertainty: the equivalent convex program. SIAM J. Appl. Math. **14**(1), 89–105 (1966)
6. Von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)
7. Markowitz, H.: Portfolio selection. J. Finance **7**(1), 77–91 (1952)
8. Yaari, M.E.: Some remarks on measures of risk aversion and on their uses. J. Econ. Theory **1**(3), 315–329 (1969). doi:10.1016/0022-0531(69)90036-2
9. Yaari, M.E.: The dual theory of choice under risk. Econometrica **55**(1), 95–115 (1987)
10. Artzner, P., Delbaen, F., Eber, J.M., Heath, D.: Thinking coherently: generalised scenarios rather than var should be used when calculating regulatory capital. Risk **10**, 68–71 (1997)
11. Artzner, P., Delbaen, F., Eber, J.M., Heath, D.: Coherent measures of risk. Math. Finance **9**(3), 203–228 (1999)
12. Rockafellar, R., Uryasev, S.: Conditional value-at-risk for general loss distributions. J. Banking Finance **26**(7), 1443–1471 (2002)
13. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. J. Risk **2**, 21–41 (2000)
14. Kusuoka, S.: On law invariant coherent risk measures. In: Advances in Mathematical Economics, pp. 83–95. Springer, Berlin (2001)
15. Bertsimas, D., Brown, D.: Constructing uncertainty sets for robust linear optimization. Oper. Res. **57**(6), 1483–1495 (2009)
16. Shapiro, A., Dentcheva, D., Ruszczynski, A.: Lectures on Stochastic Programming: Modeling and Theory. MPS-SIAM Series on Optimization. SIAM-Society for Industrial and Applied Mathematics, Philadelphia (2009)
17. Kleywegt, A., Shapiro, A., Homem-de Mello, T.: The sample average approximation method for stochastic discrete optimization. SIAM J. Optim. **12**(2), 479–502 (2002)
18. Linderoth, J., Shapiro, A., Wright, S.: The empirical behavior of sampling methods for stochastic programming. Ann. Oper. Res. **142**(1), 215–241 (2006)
19. Lim, A.E., Shanthikumar, J.G., Vahn, G.Y.: Conditional value-at-risk in portfolio optimization: coherent but fragile. Oper. Res. Lett. **39**(3), 163–171 (2011). doi:10.1016/j.orl.2011.03.004
20. Lim, C., Sherali, H.D., Uryasev, S.: Portfolio optimization by minimizing conditional value-at-risk via nondifferentiable optimization. Comput. Optim. Appl. **46**(3), 391–415 (2010)
21. Ogryczak, W., Śliwiński, T.: On solving the dual for portfolio selection by optimizing conditional value at risk. Comput. Optim. Appl. **50**, 591–595 (2011). doi:10.1007/s10589-010-9321-y
22. Künzi-Bay, A., Mayer, J.: Computational aspects of minimizing conditional value-at-risk. Comput. Manag. Sci. **3**(1), 3–27 (2006)
23. Van Slyke, R.M., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J. Appl. Math. **17**(4), 638–663 (1969)
24. Zabarankin, M., Uryasev, S.: Statistical Decision Problems, Selected Concepts and Portfolio Safeguard Case Studies, Springer Optimization and Its Applications, vol. 85. Springer, New York (2014)
25. Ogryczak, W., Śliwiński, T.: On dual approaches to efficient optimization of lp computable risk measures for portfolio selection. Asia-Pac. J. Oper. Res. **28**(01), 41–63 (2011)
26. Gay, D.M.: Electronic mail distribution of linear programming test problems. Math. Program. Soc, COAL Bull. **13**, 10–12 (1985). http://www.netlib.org/lp/. Accessed 23 Aug 2014
27. Bixby, R.E.: Solving real-world linear programs: a decade and more of progress. Oper. Res. **50**, 3–15 (2002). doi:10.1287/opre.50.1.3.17780
28. Dantzig, G.B.: Linear Programming and Extensions. Princeton landmarks in mathematics and physics. Princeton University Press, Princeton (1963)
29. Todd, M.J.: The many facets of linear programming. Math. Program. **91**(3), 417–436 (2002)
30. Gu, Z.: Private Communication (2013)

31. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W., Espinoza, D.G., Goycoolea, M., Helsgaun, K.: Certification of an optimal tsp tour through 85,900 cities. Oper. Res. Lett. **37**(1), 11–15 (2009). doi:10.1016/j.orl.2008.09.006
32. Kiwiel, K.C.: Methods of descent for nondifferentiable optimization. Lecture Notes in Mathematics, vol. 1133. Springer-Verlag, Berlin (1985)
33. Portfolio Safeguard. version 2.1. http://www.aorda.com/aod/psg.action (2009). Accessed 23 Aug 2014
34. Bienstock, D., Zuckerberg, M.: Solving lp relaxations of large-scale precedence constrained problems. Integer Programming and Combinatorial Optimization **6080**, 1–14 (2010)
35. Dantzig, G.B.: Discrete-variable extremum problems. Oper. Res. **5**(2), 266–288 (1957)
36. Gassmann, H.I.: Mslip: a computer code for the multistage stochastic linear programming problem. Math. Program. **47**(1–3), 407–423 (1990)
37. Wets, R.: Stochastic programming: Solution techniques and approximation schemes. Springer, Berlin (1983)
38. Young, M.R.: A minimax portfolio selection rule with linear programming solution. Manag. Sci. **44**(5), 673–683 (1998)
39. Konno, H., Yamazaki, H.: Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market. Manag. Sci. **37**(5), 519–531 (1991)
40. Fleming, P.J., Wallace, J.J.: How not to lie with statistics: the correct way to summarize benchmark results. Commun. ACM **29**(3), 218–221 (1986)
41. Vielma, J.P., Ahmed, S., Nemhauser, G.L.: A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. INFORMS J. Comput. **20**(3), 438–450 (2008)
42. Ledoit, O., Wolf, M.: Honey, I shrunk the sample covariance matrix. UPF Economics and Business Working Paper 691 (2003)