Noname manuscript No.
(will be inserted by the editor)

CVaR minimization by the SRA algorithm

Kolos Cs. Ágoston

Received: date / Accepted: date

Abstract Using the risk measure $CVaR$ in financial analysis has become more and more popular recently. In this paper we apply $CVaR$ for portfolio optimization. The problem is formulated as a two-stage stochastic programming model, and the SRA algorithm, a recently developed heuristic algorithm, is applied for minimizing $CVaR$.

Keywords Risk measure · CVaR · stochastic programming · numerical optimization

1 Introduction

Assessing the risk of a portfolio is an interesting problem both at the research and the application level. There are several methods to measure the risk of a portfolio. One of them is Value-at-Risk, often abbreviated as VaR. Accordingly, VaR_β denotes the maximum loss that occurs with probability β. VaR is very popular among financial analysts because it can easily be interpreted. One of the problems is that it has some theoretical and numerical drawbacks. From the theoretical viewpoint it is a disadvantage that VaR does not give any information about the magnitude of loss, if it is higher than the threshold value. It is also a problem that VaR can increase with diversification, so it does not satisfy the postulate of subadditivity (see Artzner et al., 1999). From the numerical viewpoint the problem is that minimizing VaR usually leads to nonconvex optimization problems.

It is worth mentioning that VaR type constraints (e.g. $P(\{Y \geq K\}) \geq p$) are essentially probabilistic constraints, thoroughly discussed in Prékopa

Kolos Cs. Ágoston
Corvinus University of Budapest
Tel.: +36-1-4827450
Fax: +36-1-4827430
E-mail: kolos.agoston@uni-corvinus.hu

(1995) among others. It is shown that the feasibility set determined by probabilistic constraints involving logconcavely distributed random variables is a convex set. It is also proved that some important distributions such as the nondegenerate normal distribution, the Dirichlet- and the Wishart distributions, are logconcave. Since in portfolio optimization products of the decision variables and random returns are considered, we cannot apply the above results.

Conditional Value-at-Risk ($CVaR$) addresses the previously described problems. $CVaR$ is a conditional expectation, so it takes into account the loss even if it is higher than the threshold value. It is a coherent risk measure (see Artzner et al., 1999), i.e. it is monotonic, transition-equivariant, positively homogeneous and subadditive.

We can use $CVaR$ in portfolio optimization, in particular to minimize the $CVaR$ of a portfolio. Rockafellar and Uryasev (2000) formulate this model as a linear programming problem (LP). Künzi-Bay and Mayer (2006) model the portfolio optimization as a two-stage stochastic programming problem. So we can handle the $CVaR$ portfolio optimization model with algorithms designed to solve two-stage problems.

Successive Regression Approximations (SRA) is a new heuristic algorithm for solving stochastic programming problems (not only two-stage type problems), introduced by Deák (2001, 2002, 2003, 2006). One of the features of this algorithm is that it can successfully treat large scale problems. In this paper, we apply SRA for minimizing $CVaR$.

The paper is organized as follows. In Section 2, the $CVaR$ portfolio minimization model is introduced. In Section 3, the SRA algorithm is briefly described. In Section 4, an implementation of the SRA algorithm for portfolio optimization is presented. Finally our numerical results are summarized in Section 5. The last section presents our conclusions.

2 The CVaR model

Let Y be a random variable representing the decision maker's (DM) loss (the gain is negative loss). Let $F(Y)$ denote the cumulative distribution function of Y. The risk measure VaR_β gives the threshold value

$$VaR_\beta = F^{-1}(\beta) \ , \tag{1}$$

where $F^{-1}(\cdot)$ in (1) denotes the generalized inverse of $F(y)$:

$$F^{-1}(w) = \inf_y \left\{ F(y) \geq w \right\} \ .$$

(1) means that the DM's loss is not higher than VaR_β with probability β.

The DM invests her capital in assets, so Y is the sum of random variables. In our case, the DM can choose among n assets. Let Y_i denote the future value of asset i. The DM invests amounts $x_1, x_2, ..., x_n$ in the assets. In this case, the future wealth is $\sum_{i=1}^n x_i Y_i$, so $Y = -\sum_{i=1}^n x_i Y_i$. For the sake of simplicity, let

the capital of the DM be 1, so x_i is the proportion of the capital invested in asset i, and Y_i is the return on asset i.

$CVaR$ for random variable Y gives the DM's expected loss under the condition that the loss is higher than VaR_β. For continuous random variables $CVaR$ is a conditional expected value:

$$E(Y|Y \geq VaR_\beta) = E(Y|Y \geq F^{-1}(\beta)) \ . \tag{2}$$

Rockafellar and Uryasev (2000) show that for any continuous random variable Y, $CVaR$ can be determined as

$$\min_z (z + (1-\beta)^{-1} E([Y-z]^+)) \ , \tag{3}$$

where $[x]^+$ denotes the nonnegative part of x. It is well known that VaR_β is the left endpoint of the closed interval of optimal solutions.

If Y is not a continuous random variable, then (3) is not necessarily equal to (2). The risk measure defined by (2) is not coherent. However, if we define $CVaR_\beta$ as in (3), then we get a coherent risk measure.

In our model, the DM minimizes the $CVaR$ of a portfolio contingent on her expecting at least return r^*. This model can be formulated as a two-stage stochastic programming problem (see Künzi-Bay and Mayer, 2006):

First stage:

$$\min_{\mathbf{x},z} E(Q_C(\mathbf{x}, z, \mathbf{Y}))$$

$$s.t \qquad \sum_{i=1}^{n} x_i = 1$$

$$\sum_{i=1}^{n} x_i E(Y_i) \geq r^* \ .$$

Second stage:

$$Q_C(\mathbf{x}, z, \mathbf{Y}) = z + (1-\beta)^{-1} \min_y y$$

$$s.t. \qquad y \geq -\sum_{i=1}^{n} x_i Y_i - z$$

$$y \geq 0 \ .$$

In the above problem, Y_i is a random variable representing the return on asset i, x_i is a decision variable denoting the proportion of the capital invested in asset i, z is a technical variable and β is an external parameter (the probability level for $CVaR$). We can add non-negativity constraints for x_i's, but it is not necessary to do so (short selling is allowed).

If the random variables $Y_1, Y_2, ..., Y_n$ are discrete and finite, then the two-stage problem can be solved as a linear programming problem (see Rockafellar and Uryasev, 2000)). It is a widely accepted method of solution in the literature to discretize continuous random variables (or taking samples). Künzi-Bay and Mayer (2006) give an effective algorithm for $CVaR$ minimization problems. The discretization may run into computational difficulties if the number of dimensions (number of assets) is high (see Deák, 2002).

An alternative solution technique is to apply Monte Carlo integration technique combined with the SRA method.

3 The SRA algorithm

The SRA (Successive Regression Approximations) is a recently developed heuristic algorithm (Deák, 2001, 2002, 2003, 2006) for optimization of stochastic programming models. We describe the algorithm for two-stage programming problems.

Consider a two-stage problem in the following form:

$$\begin{aligned} &\text{First stage:} \\ &\qquad \min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} + E(Q_C(\mathbf{x}, \mathbf{Z})) \\ &s.t. \qquad A\mathbf{x} = \mathbf{b} \\ &\qquad\qquad \mathbf{x} \geq \mathbf{0} \ . \\ &\text{Second stage:} \\ &\qquad Q_C(\mathbf{x}, \mathbf{Z}) = \min_{\mathbf{y}} \mathbf{q}^\top \mathbf{y} \\ &s.t. \qquad T\mathbf{x} + W\mathbf{y} = \mathbf{Z} \\ &\qquad\qquad \mathbf{y} \geq 0 \ . \end{aligned}$$

where $\mathbf{Z}$ is a random variable.

The main numerical difficulty lies in having to compute $E(Q_C(\mathbf{x}, \mathbf{Z}))$. It is easy, however, to give an unbiased estimate of $E(Q_C(\mathbf{x}, \mathbf{Z}))$. Let $\tilde{\mathbf{Z}}^1$, $\tilde{\mathbf{Z}}^2$, ..., $\tilde{\mathbf{Z}}^k$ be realizations of random variable $\mathbf{Z}$. In this case,

$$\overline{p}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} Q_C(\mathbf{x}, \tilde{\mathbf{Z}}^i) \tag{4}$$

is an unbiased estimate[1] of $E(Q_C(\mathbf{x}, \mathbf{Z}))$.

The main idea of the SRA algorithm is to compute the estimates $\overline{p}_i(\mathbf{x}^i)$ of the function value $E(Q_C(\mathbf{x}^i, \mathbf{Z}))$, construct a quadratic approximation based on $\overline{p}_i(\mathbf{x}^i)$, and then substitute a quadratic approximation for $E(Q_C(\mathbf{x}, \mathbf{Z}))$. In subsequent iterations we make the approximation more and more precise thereby approaching the optimum.

For starting the algorithm we need initial points. Usually these m initial points $(\mathbf{x}^i)$ are taken randomly, and $\overline{p}_i(\mathbf{x}^i)$ are calculated for these points. We have the set of points $S_m = \{\mathbf{x}^i, \overline{p}_i(\mathbf{x}^i)\}_{i=0}^{m-1}$, and fit a quadratic approximation to them in the form

$$q_m(\mathbf{x}) = \mathbf{x}^T D_m \mathbf{x} + \mathbf{b}_m^\top \mathbf{x} + c_m \ ,$$

where D_m, $\mathbf{b}_m^\top$ and c_m can be computed from the optimization problem:

$$\min_{D_m, \mathbf{b}_m^\top, c_m} \sum_{l=0}^{m-1} (\overline{p}_i(\mathbf{x}^i) - q_m(\mathbf{x^i}))^{\mathbf{2}} \ .$$

We substitute the following problem for the original two-stage problem

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} + q_m(\mathbf{x}) \ ,$$

[1] In applications, sometimes it is more efficient to use other methods, for details see Deák (2006)

subject to

$$A\mathbf{x} = \mathbf{b} \ ,$$

$$\mathbf{x} \geq 0 \ .$$

The we determine the optimum of the approximate problem. If the optimum is "good enough", then we stop. If not, we calculate the estimation $p(\mathbf{x})$ and add it to the previous points and fit again a quadratic approximation (the detailed description of the algorithm can be found in (Deák, 2002, 2003, 2006).

The "good enough" stopping criterion might be having achieved a required level of accuracy (see Mak, Morton and Wood, 1999).

The performance of SRA has been found efficient for solving stochastic problems but its theoretical foundation is still lacking, though a convergence proof for the one-dimensional case has been presented (Deák, 2010). In this paper, we demonstrate that the SRA algorithm can be applied for minimizing a portfolio's $CVaR$.

4 The implementation of the SRA algorithm

In Section 2, we have given the two-stage problem for minimizing a portfolio's $CVaR$. In this section, we solve this model with the SRA algorithm. During the implementation, we deviate from the original algorithm in some respects, mostly in the method of choosing the initial points.

4.1 Random technical matrices

In financial applications the so called technical matrix (T) in the second stage of the problem is typically stochastic, which is the case in our situation as well. Deák (2002) solves problems where the technical matrix is deterministic (and solves probability constrained models as well). The difficulty in handling random technical matrices is that the second stage problem may not have a solution. In our model the technical matrix is random, but the second stage problem has a solution for any realization. Thus the main ideas in the SRA algorithm can be applied for the problem we are concerned with. In this sense, our implementation can be considered as an extension as well.

4.2 Choosing initial points

First we present the method of selecting initial points. Since we know some properties of the investigated problem, we do not have to take the initial points "completely" randomly. The initial points are in the unit simplex, since $\sum_{i=1}^{n} x_i = 1$ (without the constraints $x_i \geq 0$ it is a hyperplane). The expected return constraint is satisfied as an equality in our problem (it is quite general for financial problems), so the initial points meet the condition that the expected return is r^*. Due to the beneficial effect of diversification, the optimum

is mostly somewhere in the "middle" of the simplex. So we choose the point in the simplex closest to the origin which satisfies the expected return equality as a center, and then we consider some points around it.

We have to take initial values for variable z as well. The appropriate initial values for z are essential since although the function (3) is U-shaped, it is not quadratic (it is not difficult to see that for high z values function (3)) is "almost" linear). The optimal value (left endpoint of the optimal values) of z is the VaR, which is the β-percentile of the variable. By using the normal distribution approximation or the Chebyshev inequality it is likely that we will get "appropriate" values of z.

The SRA algorithm (as described in Deák, 2002) uses all the points (including the initial points) for the quadratic approximation. This can be useful, because the "outlier" initial points help make the approximation convex, which is essential for quadratic optimization. On the other hand, in a later iteration these initial points may become "outliers", so those may hamper the algorithm in finding the exact optimum particularly if the objective function is not "exactly" quadratic. In the implementation, we choose to discard the initial points if enough points are available around the optimum.

4.3 Adding new points

It is not enough to add the new optimum to the previous points. We have to take a few other random points around it to guard against degeneracy. We use a threshold value for the possible maximal distance around the optimum.

Since we drop the initial points, sometimes we get a false optimum (the quadratic form $q_m(\mathbf{x})$ is not convex). In order to avoid this kind of unwanted points, we require that the distance between the previous and current optima do not exceed the threshold value. If we obtain a false optimum, then we "keep the direction" and substitute a closer point, which does not violate the threshold.

4.4 Stopping criteria

We use a threshold value for the difference between the approximated and the simulated $CVaR$ value. If the difference exceeds the threshold value, then the algorithm adds new random points around the previous optimum in order to make the approximation more accurate.

The algorithm stops only if the change in the optimal value is below the predefined threshold through several iterations.

Table 1 Portfolio Mean Returns

Asset	Mean return
S&P 500	0.0101110
Gov Bond	0.0043532
Small Cap	0.0137058

Table 2 Portfolio Covariance Matrix

	S&P 500	Gov Bond	Small Cap
S&P 500	0.00324625	0.00022983	0.00420395
Gov Bond	0.00022983	0.00049937	0.00019247
Small Cap	0.00420395	0.00019247	0.00764097

Table 3 Optimal Portfolio Weights

S&P 500	Gov Bond	Small Cap
0.452013	0.115573	0.432414

Table 4 $CVaR$ Values for the Optimal Portfolio

$\beta = 0,9$	$\beta = 0,95$	$\beta = 0,99$
0.096975	0.115908	0.152977

5 Numerical results

We investigate the problem described in Rockafellar and Uryasev (2000). To compare our results with theirs, we solve the problem with the SRA algorithm and with LP as well.

Rockafellar and Uryasev (2000) consider three assets: S&P 500, a portfolio of long-term U.S. government bonds (Gov Bond) and a portfolio of small capitalization stocks (Small Cap). The means and standard deviations can be found in Table 1 and 2.

We (as do Rockafellar and Uryasev, 2000) assume that the joint distribution of returns is a multivariate normal distribution. This assumption means that the optimal portfolio weights equal to those in the optimal solution of the Markowitz portfolio model (see Rockafellar and Uryasev, 2000). The optimal portfolio weights for the Markowitz model can be seen in Table 3. Table 4 shows the $CVaR$ values for the optimal portfolio.

Rockafellar and Uryasev (2000) present some numerical results but they take only one sample for each parameter set. As opposed to what they did, we repeat their process: get a random sample from the appropriate normal distribution and solve the LP problem. Then we take another random sample and solve the problem again and so on. In this way, we get a better idea (and

Table 5 Summary: the LP method (sample size, mean of $CVaR$ estimations, means of optimal assets' weight, processor time in seconds, the standard deviations can be seen in parentheses $\beta = 0.9$).

Sample size	$CVaR$	S&P 500	Gov Bond	Small Cap	Time
100	0.09251	0.38099	0.14287	0.47614	0.0
	(0.01169)	(0.26894)	(0.10337)	(0.16557)	(0.0)
500	0.09676	0.43688	0.12139	0.44173	0.0
	(0.00557)	(0.15367)	(0.05907)	(0.09461)	(0.0)
2500	0.09725	0.45195	0.11560	0.43246	0.7
	(0.00234)	(0.07267)	0.02793	0.04474	(0.1)
12500	0.09702	0.45557	0.11421	0.43023	25.7
	(0.00095)	(0.03232)	(0.01242)	(0.01990)	(2.7)

a more accurate measure) of how far the optimum thus obtained is from the true optimum.

Table 5 presents a summary of 100 samples for each parameter set. We investigate only the $\beta = 0.9$ case. In the table, the first column shows the sample size, the second shows the means of $CVaR$ estimates. It is easy to see that the $CVaR$ estimate is biased if the sample size is small. The next three columns show the means of optimal assets' weights. The sixth column contains the required running time in seconds. The standard deviations can be seen in parentheses below the means.

Rockafellar and Uryasev (2000) use the CPLEX solver, we use MINOS. We present the summary of the results in order to show that the differences are due to the different algorithms and not to the different solvers (we have used a personal computer with 2.33 GHz processor and 2 GB memory for numerical results).

The SRA algorithm is coded in Lahey Fortran. The algorithm requires unbiased estimations (random samples from the appropriate multivariate normal distribution) for the second stage problem in each iteration. Table 6 shows a summary for different parameter sets. The first column gives the sample size (k in (4)), the second is for the means of the $CVaR$ estimates. The next three columns show the means of optimal assets' weights, the sixth one presents the means of the number of iterations for the SRA algorithm, and the seventh one shows the running time in seconds. The standard deviations can be seen in parentheses below the means. The results in Table 6 demonstrate that efficient optimization can be achieved by the SRA algorithm. Small sample size examples are not included in the table, since the algorithm is not effective for small sample size.

It is worth comparing the result of SRA to the results of the LP method. The sample size has a different "meaning" in the two algorithms, so it is not meaningful to compare the results for the same sample size. It is better to compare the accuracy of the $CVaR$ estimation for the same running time.

For example we can compare the results of the LP method for sample size 12500 with the SRA algorithm for sample size 1000000. The running times are

Table 6 Summary: the SRA algorithm (sample size, mean of $CVaR$ estimates, means of optimal assets' weight, processor time in seconds, the standard deviations are in parentheses and $\beta = 0.9$).

Sample size	$CVaR$	S&P 500	Gov Bond	Small Cap	# Iteration	Time
10000	0.09697	0.46353	0.11115	0.42532	1556	2.2
	(0.00006)	(0.04028)	(0.01548)	(0.02480)	(776)	(1.1)
100000	0.09697	0.46443	0.11080	0.42477	328	4.4
	(0.00004)	(0.02796)	(0.01075)	(0.01722)	(138)	(1.8)
1000000	0.09697	0.45750	0.11346	0.42903	244	32.1
	(0.00001)	(0.01305)	(0.00502)	(0.00803)	(94)	(12.4)

close to each other. Obviously, the SRA algorithm gives a more precise result, particularly in the case of $CVaR$ estimations.

It is also worth mentioning that the SRA algorithm requires significantly less memory. Assume the sample size to be 10000. The number of the nonzero elements in the related LP matrix is about 40000. In the case of SRA, the number of nonzero elements is 6 and there are quadratic terms as well. Moreover, in SRA the number of nonzero elements does not increase with the sample size. The quadratic approximation can be determined without storing all the points, it needs only the sums of some power functions of the decision variables. So we need to store only the sums, and when we get new points only the appropriate sums should be updated.

6 Summary

$CVaR$ risk measure minimization for a given portfolio is a well known problem. It is the subject of a growing number of research papers. We have treated the problem in the framework of two-stage stochastic programming and solved it with the SRA algorithm. The SRA algorithm is a recently developed heuristic method for solving stochastic programming problems. We have demonstrated that this algorithm is suitable for $CVaR$ risk measure minimization.

We have discussed the SRA algorithm. In order to take advantage of the specialties of the problem, the original algorithm has been modified at certain places. We have also presented numerical results for the problem described in Rockafellar and Uryasev (2000). From the numerical results we can conclude that the SRA algorithm is a viable alternative for $CVaR$ minimization. The SRA algorithm is not efficient for small sample size but it works well for large samples.

Acknowledgements I am grateful to István Deák for his help in this research and preparing the manuscript. I thank Ferenc Forgó, Roger Gray, Miklós Pintér and the anonymous referees for their suggestions and remarks. Naturally, all errors are mine.

References

F. Andersson, H. Mausser, D. Rosen, S. Uryasev, Credit risk optimization with Conditional Value-at-Risk criterion, Mathematical Programming, Series B, 89, 273-291 (2001)

P. Artzner, F. Delbaen, J.-M. Eber, D. Heath, Coherent Measures of Risk, Mathematical Finance 9 no. 3, 203-228 (1999)

I. Deák, Successive regression approximations for solving equations, Pure Mathematics and Applications 12, 25-50 (2001)

I. Deák, Computing two-stage stochastic programming problems by successive regression approximations. In: Stochastic optimization techniques: Numerical Methods and Technical Applications (ed. K. Marti) Springer LNEMS V. 513, 91-102 (2002).

I. Deák, Solving stochastic programming problems by successive regression approximations -numerical results. In: Dynamic stochastic optimization (eds. K. Marti, Y. Ermoliev, G. Pflug) Springer LNEMS V.532, 209-224 (2003).

I. Deák, Two-stage stochastic problems with correlated normal variables: computational experiences, Annals of Operations Research, 142, 79-97 (2006)

I. Deák, Convergence of Succesive Regression Approximations for Solving Noisy Equations. In B.H.V. Topping, J.M. Adam, F.J. Pallarés, R. Bru, M.L. Romero, (Editors), "Proceedings of the Tenth International Conference on Computational Structures Technology", Civil-Comp Press, Stirlingshire, UK, Paper 209, (2010)

Cs. Fábián, A. Veszprémi, Algorithms for handling CVaR-constraints in dynamic stochastic programming models with applications to finance, The Journal of Risk 10, 111-131 (2006)

A. Künzi-Bay, J. Mayer, Computational aspect of minimizing conditional value-at-risk, Computational Management Science 3, 3-27 (2006)

W.-K. Mak, D. Morton, R. Wood, Monte Carlo bounding techniques for determining solution quality in stochastic programs, Operations Research Letters, Volume 24, Number 1, 47-56 (1999)

G. Pflug, Some remarks on the Value-at-Risk and the Conditional Value-at-Risk. In Probabilistic constrained optimization (ed. Uryasev), Kluwer, Dordrecht, 272-281 (2000)

A. Prékopa, Stochastic Programming. Akademiai Kiado, Kluwer, (1995)

T. Rockafellar, S. Uryasev, Optimization of Conditional Value-At-Risk, The Journal of Risk, Vol. 2, No. 3, 21-41 (2000)

T. Rockafellar, S. Uryasev, Conditional Value-at-Risk for general loss distributions, Journal of Banking & Finance 26, 1443-1471 (2002)