Quantitative Finance Letters

ISSN: 2164-9502 (Print) 2164-9510 (Online) Journal homepage: https://www.tandfonline.com/loi/rqfl20

Practical algorithms for value-at-risk portfolio optimization problems

Mingbin Feng, Andreas Wächter & Jeremy Staum

To cite this article: Mingbin Feng, Andreas Wächter & Jeremy Staum (2015) Practical algorithms for value-at-risk portfolio optimization problems, Quantitative Finance Letters, 3:1, 1-9, DOI: 10.1080/21649502.2014.995214

To link to this article: https://doi.org/10.1080/21649502.2014.995214

Quantitative Finance Letters, 2015
Vol. 3, 1–9, http://dx.doi.org/10.1080/21649502.2014.995214

Routledge
Taylor & Francis Group

Practical algorithms for value-at-risk portfolio optimization problems

MINGBIN FENG*, ANDREAS WÄCHTER and JEREMY STAUM

Northwestern University, USA

(*Received 11 February 2014; in final form 14 November 2014*)

This article compares algorithms for solving portfolio optimization problems involving value-at-risk (VaR). These problems can be formulated as mixed integer programs (MIPs) or as chance-constrained mathematical programs (CCMPs). We propose improvements to their state-of-the-art MIP formulations. We also specialize an algorithm for solving general CCMPs, featuring practical interpretations. We present numerical experiments on practical-scale VaR problems using various algorithms and provide practical advice for solving these problems.

Keywords: value-at-risk; portfolio optimization; mixed integer programming; mixed integer linear programming; chance-constrained mathematical programs

JEL Classification: C02; C61; G11

1. Introduction

Value-at-risk (VaR) and conditional value-at-risk (CVaR) are both risk measures that describe tail risks of financial portfolios. VaR can be defined as a percentile of the loss distribution and CVaR as the average loss for losses exceeding the corresponding VaR. Readers are encouraged to refer to Rockafellar and Uryasev (2002) for rigorous discussions on VaR and CVaR for general distributions. The study of coherent risk measures (Artzner *et al.* 1999) and the development of a convex CVaR optimization framework (see Krokhmal *et al.* 2002, Rockafellar and Uryasev 2002, and references therein) show advantages of CVaR over VaR. In particular, scenario-based CVaR problems can be reformulated as linear programs (LPs). Even when the number of assets and the number of scenarios are large, there are special formulations and algorithms that can solve CVaR problems more efficiently than the originally proposed LP reformulation (see Hong and Liu 2009, Lim *et al.* 2010, Ogryczak and Śliwiński, 2011, for example). Nevertheless, since its introduction as a risk measure VaR has played an important role in both practical applications and academic research (see Jorion 1997, Pritsker 1997, for example). In particular, VaR has been written into industry regulations such as Basel Accord II (BCBS 2004) as a capital adequacy requirement. Despite the aforementioned favorable developments for CVaR, VaR retains a similar status in Basel Accord III (BCBS 2010), which is scheduled to be implemented from 2013 to 2018. In addition to its practical importance, VaR also attracts significant academic research attention. For example, Kou and Peng (2012) provide counterarguments against common criticisms, and Kou *et al.* (2013) recommend a new risk measure for regulatory purpose, the tail conditional median, which is essentially VaR at a higher confidence level. As a consequence, VaR optimization both is a valuable tool in practice and an interesting area of research.

VaR problems can be solved heuristically or optimally. Readers are encouraged to refer to Gilli and Këllezi (2002), Larsen *et al.* (2002), and Hong *et al.* (2014) for various heuristics for VaR problems. In this article we explore ways to solve VaR problems to global optimality. Since VaR is defined as a percentile of the portfolio loss distribution, optimization problems with a VaR constraint or objective can be naturally formulated as chance-constrained mathematical programs (CCMPs). A common approach to solve CCMPs is to reformulate them into mixed integer programs (MIPs). Two MIP formulations for VaR portfolio optimization problems are proposed by Benati and Rizzi (2007) and are commented on by Lin (2009). Moreover, there have been advances in recent years for solving general CCMPs directly (see Qiu *et al.* 2012, Song *et al.* 2012, Luedtke 2013, for example). We examine both the traditional approach and the new advances and compare their effectiveness in solving practical VaR problems.

This paper is organized as follows. Section 2 reviews the definition of VaR and presents common MIP formulations for VaR portfolio optimization problems. Section 3 proposes some improvements for the MIP formulations. The application of a decomposition algorithm for general CCMPs to VaR problems is shown in Section 4. Section 5 summarizes results of numerical experiments. Section 6 concludes with practical advice for solving large-scale VaR problems.

2. VaR problems and MIP formulations

Given a universe of n assets and a set of m scenarios, denote the loss (or the negative return) incurred in scenario i by per unit of asset j by L_{ij}. We denote the loss matrix by

*Corresponding author. Email: mbfeng@u.northwestern.edu

$L = [L_{ij}]$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$. Each row of the loss matrix, $L_{i\cdot}$, represents the assets' losses in scenario i, $i = 1, \ldots, m$. We denote the probability of realizing scenario i by p_i, $i = 1, \ldots, m$ and the decision variable by $x \in \mathcal{X}$, where $\mathcal{X}$ is the set of all feasible portfolios. We assume that the loss of asset j in any scenario is proportional to $x_j, j = 1, \ldots, n$. Then the portfolio loss random variable is represented by the vector $l = l(x, L) = Lx$, where $l_i = L_{i\cdot}x$ represents the portfolio loss in the ith scenario.

Given $\alpha \in [0, 1]$, VaR at confidence level α, or α-VaR, of the portfolio loss random variable l is defined as its α-percentile. Rigorous definitions of α-VaR for general distributions can be found in Rockafellar and Uryasev (2002). In our settings the α-VaR is computed as follows.

PROPOSITION 2.1 *Assume that for each portfolio $x \in \mathcal{X}$ the distribution of the portfolio loss $l(x, L)$ is concentrated in finitely many points, that is, the vector $l = Lx$ has finite dimension. Let those losses be ordered $l_{(1)} \geq l_{(2)} \geq \cdots \geq l_{(m)}$ and let the corresponding probabilities be $p'_i = \Pr(l = l_{(i)}) > 0$. The α-VaR of the loss is given by $\zeta_\alpha(x) := l_{(m_\alpha+1)}$ where m_α is the unique index such that $\sum_{i=1}^{m_\alpha} p'_i < 1 - \alpha \leq \sum_{i=1}^{m_\alpha+1} p'_i$.*

To simplify exposition we assume that all scenarios are equally probable, that is, $p_i = 1/m$ for $i = 1, \ldots, m$. Then the unique index m_α in Proposition 2.1 is given by $m^* = \lfloor (1 - \alpha)m \rfloor$. The extension to general discrete distributions is straightforward. We are interested in finding a portfolio with minimum α-VaR of portfolio losses among all feasible portfolios, that is,

$$\min\{\zeta_\alpha(x) | x \in \mathcal{X}\}. \tag{1}$$

The set of feasible portfolios $\mathcal{X}$ is described by a given set of portfolio constraints. Examples of portfolio constraints considered in practice are budget constraint ($\sum_{i=1}^{n} x_j = 1$), no short-selling constraints $x_j \geq 0$, $j = 1, \ldots, n$, etc. One can formulate (1) into a mathematical program using implication constraints, which requires additional binary variables and an auxiliary variable. In particular, by introducing binary variables $b \in \{0, 1\}^m$ and an auxiliary variable $z \in \mathbb{R}$ we can reformulate (1) into the following mathematical program with implication constraints.

$$\begin{aligned}
\text{minimize} \quad & z && \text{(2a)}\\
\text{s.t.} \quad & b_i = 0 \Rightarrow L_{i\cdot}x \leq z, \quad \forall i = 1, \ldots, m, && \text{(2b)}\\
& \sum_{i=1}^{m} b_i \leq m^*, \quad \text{and} && \text{(2c)}\\
& (x, b, z) \in \mathcal{X} \times \{0, 1\}^m \times \mathbb{R}. && \text{(2d)}
\end{aligned}$$

Constraints (2b) and (2c) ensure that $\Pr\{l(x) \leq z\} \geq \alpha$, so for any feasible solution (x, b, z) the value of z is an upper bound for $\zeta_\alpha(x)$. Moreover, since $z \in \mathbb{R}$ and (2) is a minimization problem, if (x^*, b^*, z^*) is an optimal solution, then z^* must be the least upper bound for $\zeta_\alpha(x^*)$, which is $\zeta_\alpha(x^*)$ itself. In particular, z^* equals the $(m^* + 1)$th largest portfolio loss, that is, $z = L_{i^*\cdot}x^*$ for some scenario i^*. If the probability of realizing each scenario is different, that is, $\Pr\{l(x) = L_{i\cdot}x\} = p_i$, $i = 1, \ldots, m$, the only change needed is replacing (2c) by $\sum_{i=1}^{m} p_i b_i \leq \alpha$.

A common way to solve (1) is to reformulate (2) into a MIP using the so-called 'big-M' constraints given by

$$L_{i\cdot}x - M_i b_i \leq z, \quad \forall i = 1, \ldots, m. \tag{3}$$

If one replaces (2b) with (3), then the MIP formulation for (1) is obtained, provided that the coefficient vector $M \in \mathbb{R}^m$ is sufficiently large. It is clear that for any $M_i \in \mathbb{R}$, $L_{i\cdot}x \leq z$ if $b_i = 0$. If $b_i = 1$, then (2b) does not impose any restriction on the variables (x, b, z). Thus if $M_i \in \mathbb{R}$ is large enough to ensure the constraint $L_{i\cdot}x - M_i \leq z$ is not active when $b_i = 1$, then (3) is a valid reformulation of (2b). However, big-Ms that are too large result in poor continuous relaxations in branch-and-bound methods and hence hinder computational performance. Suitable choices of M_i, $i = 1, \ldots, m$ for constraints (3) is the main focus in the next section.

3. Big-Ms tailored for VaR problems

In general, finding suitable big-Ms for MIP formulations can be a challenging task. For portfolio optimization problems such as (1) there is often a natural way of bounding the portfolio losses using bounds of individual assets' losses, which can help calculating big-Ms. For example, to deactivate constraint (3) it requires $M_i \geq L_{i\cdot}x^* - z^* = L_{i\cdot}x^* - L_{i^*\cdot}x^*$ for some scenario i^*. Without knowing the optimal solution an upper bound on the difference of scenario losses is needed. Knowing the bounds of portfolio losses $l^- \leq Lx \leq l^+$ for some $l^- \leq l^+ \in \mathbb{R}^m$, then $M_i = l_i^+ - l_i^-$, $i = 1, \ldots, m$ is valid for (3).

Due to limited liability, common stocks have a maximum loss of 100%. The big-Ms considered by Benati and Rizzi (2007) are based on this bound. Similarly, one can derive natural bounds for an insurance portfolio or any long-only credit portfolio because individual losses in these cases are non-negative. Prior to the developments in chance-constrained mathematical programming in the last few years (see Qiu *et al.* 2012, Luedtke 2013, among others), this natural way of calculating big-Ms was the state-of-the-art approach for solving VaR problems as MIPs. We present a different choice of big-Ms that is tailored to VaR problems. This choice entails additional computational costs but we expect it to improve the overall computational efficiency. The following discussions about big-Ms not only reveal a choice of big-Ms for VaR problems but also provide meaningful interpretations for a specialized decomposition algorithm, which is presented in the next section.

DEFINITION 3.1 *For any two scenarios i, j, define the relative excessive loss (REL) of i relative to j as*

$$d_j(i) = \max\{(L_{i\cdot} - L_{j\cdot})x | x \in \mathcal{X}\}. \tag{4}$$

Consider the following *pivoting procedure*: For a scenario $i \in \{1, \ldots, m\}$, called the *pivot scenario*, we calculate

$d_j(i)$ for all $j = 1, \ldots, m$, and then sort them to obtain a permutation $\sigma(i)$ of $\{1, \ldots, m\}$ such that

$$d_{\sigma_1(i)}(i) \leq d_{\sigma_2(i)}(i) \leq \cdots \leq d_{\sigma_m(i)}(i). \tag{5}$$

The following proposition shows how the RELs and the pivoting procedure relate to the feasible set of (2). A valid choice of big-Ms for constraints (3) is presented based on these relationships.

PROPOSITION 3.2 *Given any pivot scenario* $i \in \{1, \ldots, m\}$, *the following constraints are valid for any feasible solution* (x, b, z) *to problem* (2).

$$L_{i\cdot}x - d_{\sigma_{m^*+1}(i)}(i) \leq z \tag{6a}$$

$$b_{\sigma_k(i)} = 0 \Rightarrow L_{i\cdot}x - d_{\sigma_k(i)}(i) \leq z, \quad \forall k = 1, \ldots, m \tag{6b}$$

$$\begin{cases} L_{i\cdot}x - d_{\sigma_k(i)}(i) \leq z, & \text{if } b_{\sigma_k(i)} = 0, \quad k = 1, \ldots, m^* \\ L_{i\cdot}x - d_{\sigma_{m^*+1}(i)}(i) \leq z, & \text{otherwise.} \end{cases} \tag{6c}$$

A formal proof for Proposition 3.2 is given in Appendix 1. We present here the intuitive interpretations for constraints (6). By definition of α-VaR in our setting there can be at most m^* scenarios with positive excessive losses over α-VaR. However, for any $x \in \mathcal{X}$ and any pivot scenario i, by definition of RELs and the pivoting procedure there are at least $m^* + 1$ scenarios whose losses are greater than or equal to $L_{i\cdot}x - d_{\sigma_{m^*+1}(i)}(i)$. Therefore α-VaR must be at least $L_{i\cdot}x - d_{\sigma_{m^*+1}(i)}(i)$, which validates constraint (6a). If $b_{\sigma_k(i)} = 0$, the $\sigma_k(i)$th scenario loss is less than or equal to α-VaR. Therefore the difference between the ith scenario loss and the α-VaR cannot exceed that between the ith scenario's and the $\sigma_k(i)$th scenario's loss, whose maximum is $d_{\sigma_k(i)}(i)$. The above interpretations remain valid for any upper bound of α-VaR instead of α-VaR. For any feasible solution (x, b, z) to (1), z is an upper bound of α-VaR for portfolio x thus constraints (6a) and (6b) are valid. Based on Equation (6a) we propose the following big-Ms.

PROPOSITION 3.3 *With the following big-Ms, Equation* (3) *are valid constraints for Equation* (2):

$$M_i = d_{\sigma_{m^*+1}(i)}(i) - d_i(i) = d_{\sigma_{m^*+1}(i)}(i), \quad \forall i = 1, \ldots, m, \tag{7}$$

where the second equality holds because $d_i(i) = \max\{(L_{i\cdot} - L_{i\cdot})x | x \in \mathcal{X}\} = 0$.

Proof For any scenario i, if $b_i = 0$ then Equation (3) requires that $L_{i\cdot}x \leq z$. If $b_i = 1$, then Equation (3) becomes Equation (6a). Because Equation (6a) is valid for any feasible solution to (2), the big-M constraint (3) is not active. Therefore, Equation (7) is valid because this choice of big-M makes Equation (3) a valid reformulation of Equation (2b). ■

In general, computing Equation (7) requires solving m^2 optimization problems. Efficient algorithms for solving these problems, if available, should always be considered. For example, if $\mathcal{X}$ has one linear constraint and bound constraints then Equation (4) is a continuous knapsack problem with one constraint, which can be solved efficiently with an $O(n \log n)$ algorithm. Suppose $\mathcal{X}$ is described by a small number of N constraints. In general, Equation (4) has n variables and N constraints, which is usually a small optimization problem. Qiu *et al.* (2012) propose an iterative algorithm to tighten the big-Ms. In that proposal each iteration of the algorithm requires solving m optimization problems with $m + n$ variables and $m + N$ constraints, which may be computationally expensive in financial applications where $n \ll m$ and $N \ll m$.

A decomposition algorithm is presented in the next section. We will see the similarities between constraints (6b) and (6c) and the valid inequalities used in the decomposition algorithm in the following discussions.

4. A branch-and-cut decomposition algorithm for VaR problems

Note that Equation (1) can be viewed as a special case of CCMP because it can be cast as

$$\min\{z | Pr\{Lx - z \leq 0\} \geq \alpha, \ (x, z) \in \mathcal{X} \times \mathbb{R}\}. \tag{8}$$

In this section, we apply the decomposition algorithm proposed by Luedtke (2013) for general CCMPs to solve Equation (8). Some components of the algorithm can be simplified due to the VaR problems' special structures.

The outer loop of decomposition algorithm proposed by Luedtke (2013) is similar to a branch-and-bound method. A generic branch-and-bound method for solving MIPs can be summarized as follows. At the beginning, a continuous relaxation of a master problem is solved where all integrality constraints on variables are relaxed. This initial master problem is commonly referred to as the *root node*. Two *child nodes* are then created, which inherit the master problem and one of the constraints $b_i \leq \hat{b}_i^-$ or $b_i \geq \hat{b}_i^+$ where b_i is an integral variable but has a fractional optimal value $\hat{b}_i$ at the current node and $\hat{b}_i^-$ (respectively, $\hat{b}_i^+$) is the largest (respectively, smallest) integer that is less than (respectively, greater than) $\hat{b}_i$. Based on the optimal solution of a child node, new child nodes are created. In general, a node is fathomed when it is infeasible, when all discrete variables happen to have optimal integer values, or when the optimal objective is worse than the best optimal objective from all previously found integral solutions. A fathomed node will not be processed any further. A node is open if it has not been fathomed yet. The branch-and-bound method terminates when there are no open nodes.

As stated by Luedtke (2013), the only important difference between the decomposition algorithm and a standard branch-and-bound algorithm is how nodes are processed. The decomposition algorithm generates valid inequalities based on optimal solutions to current node. The current node is solved repeatedly if a valid inequality is violated and added to the current node. The initial master problem for the decomposition algorithm may not be a complete description of the original chance-constrained problem so it is possible to generate valid inequalities even when the

solution to the current node is integral. For ease of discussion, if the solution to the current node is integral, then the valid inequalities generated with respect to this solution are referred to as *lazy constraints*. Otherwise when the solution to current node is fractional then the valid inequalities generated are called *user cuts*. This terminology is consistent with those in the CPLEX documentation (IBM ILOG 2013).

Applying the decomposition algorithm proposed by Luedtke (2013) to solve Equation (8) requires solving a master problem of the following form:

$$MP(N_0, N_1, C) := \min \quad z \tag{9a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} b_i \leq m^*, \quad b \in [0,1]^m \tag{9b}$$

$$(x, b, z) \in C, \quad (x, z) \in \mathcal{X} \times \mathbb{R} \tag{9c}$$

$$b_i = 0, i \in N_0, \quad b_i = 1, i \in N_1, \tag{9d}$$

where $C \subseteq \mathbb{R}^{n+m+1}$ is a polyhedron that contains the feasible region of Equation (2), and $N_0, N_1 \subseteq \{1, \ldots, m\}$ are such that $N_0 \cap N_1 = \emptyset$. Note that the description of C may not contain all implication constraints at the beginning, but it will be augmented when processing the branch-and-bound tree so that correctness of the algorithm is guaranteed.

The following components are needed to fully specify our decomposition algorithm:

(i) Given a solution to the current node, identify pivot scenarios from which violated valid inequalities may be obtained.
(ii) Solve subproblems that are required to generate valid inequalities.
(iii) Identify a good violated inequality from the set of inequalities generated from step (ii).

Component (i) can be specified in a straightforward manner. As shown by Luedtke (2013), given a solution $(\hat{x}, \hat{b}, \hat{z})$ to the current node, any scenario whose implication constraint is violated can be considered to generate valid inequalities. For our problem specifically, we consider a scenario i as a pivot scenario if $\hat{b}_i = 0 \text{and} L_i.\hat{x} > \hat{z}$. There may be multiple scenarios that satisfy this condition. Luedtke (2013) shows that if the candidate solution is not feasible for Equation (2), then for any of these scenarios at least one valid inequality, which will be presented below, is violated and can be added to C.

To specify component (ii), suppose scenario i is chosen to be a pivot scenario. If one follows the steps of the general decomposition algorithm in the original proposal (Luedtke 2013) then the required single-scenario subproblems for scenario i are $h_j(i) = \min\{z - L_i.x | z - L_j.x \geq 0, (x, z) \in \mathcal{X} \times \mathbb{R}\}$ for all $j = 1, \ldots, m$. We suggest the following simplifications to these subproblems:

$$\begin{aligned} h_j(i) &= \min\{z - L_i.x | z - L_j.x \geq 0, \ (x, z) \in \mathcal{X} \times \mathbb{R}\} \\ &= \min\{L_j.x - L_i.x | x \in \mathcal{X}\} = -d_j(i) \end{aligned} \tag{10}$$

because $z \in \mathbb{R}$ and appears in only one constraint $z - L_j.x \geq 0$. Our suggestion helps to reduce one variable and one constraint in every subproblem. This reduction not only helps to solve the subproblems more efficiently but also lends the subproblems practical interpretations in the context of VaR-optimization, as discussed in Section 3.

To condense notation, the dependence on pivot scenario i for $d_j(i)$ and $\sigma(i)$ is suppressed in the following discussion. After pivoting on scenario i (i.e. calculating its RELs and sorting them) the following inequalities

$$L_i x - z - (d_{\sigma_{m^*+1}} - d_{\sigma_k}) b_{\sigma_k} \leq d_{\sigma_k}, \quad k = 1, \ldots, m^* \tag{11}$$

are valid for the feasible set of Equation (2). Inequalities (11) can be viewed as an application of Lemma 1 of Luedtke (2013) to the feasible set of Equation (2). Alternatively, they can also be seen as a way to formulate both Equations (6a) and (6b) in one inequality for scenarios $\sigma_1, \ldots, \sigma_{m^*}$. In particular, if $b_{\sigma_k} = 1$, then Equation (11) resembles Equation (6a). If $b_{\sigma_k} = 0$, then Equation (11) resembles Equation (6b). The following theorem presents a family of valid inequalities by mixing Equation (11). The valid inequalities generated in component (ii) are presented in the following theorem.

THEOREM 4.1 *For any pivot scenario i, let $R = \{\rho_1, \ldots, \rho_{j^*}\} \subseteq \{\sigma_1, \ldots, \sigma_{m^*}\}$ be any non-empty index set such that $d_{\rho_j} \leq d_{\rho_{j+1}}$ for $j = 1, \ldots, j^*$, where $d_{\rho_{j^*+1}} = d_{\sigma_{m^*+1}}$. Then the inequalities*

$$L_i.x - z - \sum_{j=1}^{j^*} [d_{\rho_{j+1}} - d_{\rho_j}] b_{\rho_j} \leq d_{\rho_1} \tag{12}$$

are valid for the feasible set of Equation (2).

Theorem 4.1 is a special case of Theorem 3 of Luedtke (2013) and readers are encouraged to refer to his discussion on the strength of these inequalities. It is beneficial to have $\rho_1 = \sigma_1$ in Equation (12) because in that case the inequalities are facet-defining in a certain sense (Luedtke 2013). Moreover, setting $R = \{\sigma_1, \sigma_k\}$ for some $k \in \{2, \ldots, m^*\}$ yields the inequality

$$L_i.x - z - (d_{\sigma_{m^*+1}} - d_{\sigma_k}) b_{\sigma_k} - (d_{\sigma_k} - d_{\sigma_1}) b_{\sigma_1} \leq d_{\sigma_1}, \tag{13}$$

which dominates inequality (11) for this particular k. We observe that by setting $R = \{i\}$ inequalities (12) resemble the form of big-M constraints (3) with big-Ms defined in Equation (7). Therefore the specialized decomposition algorithm is able to generate big-M constraints when needed.

It is shown by Luedtke (2013) that constraints (12) are both a necessary and sufficient description for the implication constraints (2b). The discussion on big-Ms in Section 2 sheds some light on the interpretations of Equation (12). They are a compact way of expressing the necessary conditions (6c). Specifically, any (x, b, z) that satisfies Equation (12) also satisfies Equation (6c).

Lastly, we specify component (iii). For each pivot scenario i there are exponentially many $(2^{m^*} - 1$ to be

precise) valid inequalities of the form (12). Separating a 'good' valid inequality from this exponential family is the next task. An $O(m^* \log m^*)$ separation algorithm is proposed by Günlük and Pochet (2001) and is employed in Luedtke (2013). We show that the complexity of the separation algorithm proposed by Günlük and Pochet (2001) improves from $O(m^* \log m^*)$ to $O(m^*)$ when it is implemented as a subroutine of the decomposition algorithm. This is not only true in our VaR problem but also true in general when implementing the decomposition algorithm in Luedtke (2013) to solve CCMPs.

We first formulate the separation problem mathematically. Adding $d_{\sigma_{m^*+1}} = d_{\rho_{j^*+1}}$ to both sides of Equation (12) we obtain $(L_i.x - z - d_{\sigma_{m^*+1}}) - \sum_{j=1}^{j^*}(d_{\rho_{j+1}} - d_{\rho_j})b_{\rho_j} \leq (d_{\rho_1} - d_{\rho_{j^*+1}})$. Then we have $(L_i.x - z - d_{\sigma_{m^*+1}}) - \sum_{j=1}^{j^*}(d_{\rho_{j+1}} - d_{\rho_j})b_{\rho_j} \leq \sum_{j=1}^{j^*}(d_{\rho_j} - d_{\rho_{j+1}})$ by replacing the RHS with a telescopic sum. Finally, we group the like terms and arrive at the following inequality:

$$L_i.x - z - d_{\sigma_{m^*+1}} \leq \sum_{j=1}^{j^*}(d_{\rho_j} - d_{\rho_{j+1}})(1 - b_{\rho_j}). \tag{14}$$

Although Equation (14) is algebraically equivalent to Equation (12), its LHS is independent of the choice of the index set R. Given a candidate solution $(\hat{x}, \hat{b}, \hat{z})$ to the current node, that is, $\hat{b}$ may be fractional, one can minimize the RHS of Equation (14) to identify a most violated inequality in Equation (12), if any, or determine that none can be found. The minimization problem to be solved is as follows.

$$\min_{\rho_1,\ldots,\rho_{j^*}} \quad \sum_{j=1}^{j^*}(d_{\rho_j} - d_{\rho_{j+1}})(1 - \hat{b}_{\rho_j}) \tag{15a}$$

$$\text{s.t.} \quad \{\rho_1, \ldots, \rho_{j^*}\} \subseteq \{\sigma_1, \ldots, \sigma_{m^*}\}, \quad \text{and} \tag{15b}$$

$$d_{\rho_j} \leq d_{\rho_{j+1}}, \quad \forall j = 1, \ldots, j^*. \tag{15c}$$

Next we examine optimal solutions for Equation (15). Observe that the index set $R' = R \cup \{\sigma_1\}$ for any feasible R is also feasible and has objective value no greater than that of R. Therefore there is at least one optimal solution in which σ_1 is a member. Following the arguments in Günlük and Pochet (2001) we can find an optimal index set $R^* = \{\rho_1, \ldots, \rho_{j^*}\}$ which satisfies the conditions $\rho_1 = \sigma_1$ and $\hat{b}_{\rho_j^*} < \hat{b}_{\rho_{j^*-1}} < \cdots < \hat{b}_{\rho_1}$. An $O(m^*)$ algorithm for identifying such an index set is shown in lines 6 to 9 in Algorithm 2 in Appendix 2.

We state a basic version of the specialized algorithm for solving VaR problems in Algorithm 1 in the appendix. The algorithm is a specialization of the one proposed by Luedtke (2013) with a detailed separation subroutine Algorithm 2. The general mechanism of the algorithm can be found in its original proposal and is omitted to avoid repetition. The user cuts added (Line 17 of Algorithm 1) do not affect the correctness of the algorithm but may be used to improve computational efficiency. For example, one can include user cuts every other 50 nodes in the branch-and-bound tree or include only user cuts that have significant violations. In addition, this decomposition algorithm can be used in conjunction with the special branching scheme proposed by Qiu *et al.* (2012). The special branching scheme adds a cut of the form $L_i.x \geq z$ to nodes that have $i \in N_1$ and all subsequent child nodes. Such local cuts enforce the implication $b_i = 1 \Rightarrow L_i.x \geq z$ which does not affect the correctness of the decomposition algorithm. Readers are encouraged to refer to Qiu *et al.* (2012) for details of the special branching scheme.

5. Numerical experiments

Numerical experiments are conducted using CPLEX 12.5.1 on a desktop running Ubuntu 12.04.4 LTS, with 8 cores 3.4 GHz Intel Core i7, 32 GB RAM and 32 GB swap space. We set the number of threads to one for fair comparison. Some advanced CPLEX callbacks are necessary to implement the decomposition algorithm and the special branching scheme described at the end of Section 4. However, some advanced CPLEX features, such as dynamic search, are disabled when advanced callbacks are used. Unless stated otherwise, advanced CPLEX features are disabled for fair comparison.

We considered VaR-minimization problems of the form $\min\{\zeta_\alpha(x) | a \leq x \leq b, cx \geq c^*\}$ for reinsurance portfolios. An anonymous reinsurance company generously provided us with simulated reinsurance loss and premium data. The data was hypothetically simulated to represent realistic risk and returns of real business. The characteristics of these loss matrices include non-negativity, sparsity (e.g. 8% to 30% non-zero entries), positive skewness, and high kurtosis. It is shown by Qiu *et al.* (2012) that sparsity is favorable for computation efficiency. The premiums c_j, $j = 1, \ldots, n$ are the market premiums of the risk contracts. The decision variable x_j, $j = 1, \ldots, n$ represents the fraction of risk contract j that the reinsurer is willing to take. The portfolio premium is then cx. A reward constraint $cx \geq c^*$ impose a minimum level of portfolio premium. In our experiments we set $a = 0$, $b = 1$, and $c^* = 0.1 \sum_{j=1}^{n} c_j$.

In our first set of experiments we compare the following algorithms:

(i) *NaturalM*: MIP formulation with a natural choice of big-Ms. Since reinsurance losses $L \geq 0$ and the decision variables $0 \leq x \leq 1$, the portfolio loss in scenario i is bounded by $0 \leq L_i.x \leq \|L_i.\|_1$, $i = 1, \ldots, m$. The discussion at the beginning of Section 2 explains that $M_i = \|L_i.\|_1 = \sum_{j=1}^{n} |L_{ij}|$, $i = 1, \ldots, m$ is a natural choice of big-Ms.

(ii) *TightM*: MIP formulation with big-Ms defined in Equation (7). In our experiments $\mathcal{X}$ contains only one linear constraint and bound constraints. Therefore, the REL subproblems can be solved with a $O(n \log n)$ algorithm.

(iii) *Decomp*: Decomposition algorithm with lazy constraints only. The initial master problem has an empty description in the cut set C. This is a minimal form of the decomposition algorithm because no user cut is added. When a strong valid inequality of the form (12) is violated by a integer solution

to the current node, we add not only this inequality but also any inequality of the form (13) that is violated. As observed by Luedtke (2013), including inequalities (13) is a simple way to add additional sparse valid inequalities in one round.

(iv) *Decomp* + : Decomposition algorithm with lazy constraints as well as user cuts at the root node. At the root node we continue generating valid inequalities with violation greater than 0.01 until none can be found.

(v) *TightMB*: TightM with special branching scheme implemented, as discussed at the end of Section 4.

(vi) *DecompB*: Decomp with special branching scheme implemented.

We minimize the 99.5%-VaR for 10 instances of reinsurance problems with 5000 scenarios and 25 risk contracts. A time limit of 3 hours for each instance is set, which is a realistic computational budget. We are interested in both the CPU time if an instance is solved and the final optimality gap if an instance is halted due to the time limit.

Table 1 summarizes the CPU time of the 10 instances using different algorithms. We define the optimality gap as $(\text{Obj}^{\text{inc}} - \text{Obj}^{\text{lb}})/\text{Obj}^{\text{inc}}$ where Obj^{inc} is the objective value of best incumbent found up to the current node and Obj^{lb} is the best lower bound. Table 2 summarizes the optimality gaps after 3 hours if an instance is not solved within the time limit. NaturalM serves as a benchmark for the comparison because it is the state-of-the-art method. We see from tables 1 and 2 that TightM, TightMB, Decomp, and Decomp + all have better performance than NaturalM: More instances are solved within the time limit, it takes less CPU time to solve an instance, and there is a smaller optimality gap when a problem is halted. It takes only about 20 seconds to solve $5000^2 = 2.5 \times 10^7$ subproblems to calculate the tight big-Ms. Recognizing the knapsack problem structure enables us the calculate the tight big-Ms so efficiently. Our experiments show that TightM and TightMB are the fastest algorithms to solve a given instance within time limit. On average, TightM and TightMB can solve an instance more than 20 times faster than the NaturalM does. However, since this observation takes into account only problems that are solved within an arbitrary time limit, it may be an overstatement about the improvement of TightM and TightMB over NaturalM. For instances which are halted, either TightM, TightMB, or Decomp + has the smallest optimality gap. In 1 out of 10 instances Decomp takes less CPU time than TightM does and Decomp + has a smaller optimality gap than TightM does.

The addition of a special branching scheme to TightM yields little additional improvement in either CPU time or optimality gap, as evidenced by comparing columns TightM and TightMB in tables 1 and 2. We also observe a significant deterioration in CPU time and optimality gap when the special branching scheme is implemented in conjunction with Decomp when comparing columns Decomp and DecompB in tables 1 and 2. Even when DecompB manages to solve an instance, its CPU time is much longer than that of Decomp. We currently have no explanation why the special branching scheme deteriorates the decomposition algorithm so significantly.

For NaturalM and TightM, it is for the sake of fair comparison that the advanced CPLEX features were disabled. In practice, however, these features should be enabled

Table 1. CPU time (in seconds) for solving 99.5%-VaR minimization problems with 5000 scenarios and 25 risk contracts (including time for calculating big-Ms).

Instance ID	NaturalM	TightM	TightMB	Decomp	Decomp +	DecompB
1	(–)	1135.0	1080.6	(–)	3480.8	(–)
2	(–)	(–)	(–)	(–)	(–)	(–)
3	(–)	(–)	(–)	(–)	(–)	(–)
4	7035.6	29.1	29.0	168.8	97.7	2589.7
5	1699.6	45.1	45.5	150.8	175.0	(–)
6	(–)	473.5	465.0	1649.5	1439.2	(–)
7	(–)	3232.9	2775.9	2686.7	3506.8	6320.7
8	4757.7	470.7	475.7	934.2	797.3	(–)
9	(–)	(–)	(–)	(–)	(–)	(–)
10	(–)	(–)	(–)	(–)	(–)	(–)

Table 2. Optimality gap after 3 hours for solving 99.5%-VaR minimization problems with 5000 scenarios and 25 risk contracts.

Instance ID	NaturalM	TightM	TightMB	Decomp	Decomp +	DecompB
1	23.69%	(–)	(–)	0.79%	(–)	16.91%
2	24.16%	4.97%	5.39%	6.86%	5.33%	7.55%
3	43.40%	16.02%	13.92%	13.34%	10.78%	33.61%
4	(–)	(–)	(–)	(–)	(–)	(–)
5	(–)	(–)	(–)	(–)	(–)	0.50%
6	11.56%	(–)	(–)	(–)	(–)	7.84%
7	16.45%	(–)	(–)	(–)	(–)	(–)
8	(–)	(–)	(–)	(–)	(–)	5.72%
9	23.52%	7.28%	6.68%	10.90%	8.60%	24.53%
10	18.35%	0.87%	1.63%	5.72%	2.29%	25.06%

for NaturalM and TightM. The second to fifth columns of table 3 summarize the CPU time and final optimality gap for the same 10 instances when CPLEX features are enabled (CPLEX default settings). We see that enabling CPLEX features significantly improves performance for both NaturalM and TightM. More problems are solved within the given time limit and problems are solved much faster. Under the CPLEX default settings TightM can be 2–14 times faster than NaturalM. When a problem is halted, TightM always has a smaller optimality gap than that of NaturalM. With TightM, 9 out 10 instances have final optimality gaps less than 3%. Compared to NaturalM with CPLEX features disabled, the computational enhancement from calculating the tight big-Ms is more significant than enabling the CPLEX features. Comparing the second column in table 3 and the third column (TightM) in table 1 we see that TightM with CPLEX features disabled takes less CPU time to solve an instance than NaturalM with the features enabled.

The algorithms can find a high-quality solution quickly but devote most of the computation time to proving optimality. We define the incumbent gap as $(\text{Obj}^{\text{inc}} - \text{Obj}^*)/\text{Obj}^*$ where Obj^{inc} and Obj^* are the objective of the current best incumbent and of the optimal solution, respectively. Table 4 shows incumbent gaps for various algorithms after 1 minute and 10 minutes. Although solving VaR-minimization problems of the size we considered may take hours or even days, as shown in the fourth column of table 3, the algorithms can find a high-quality feasible solution in a few minutes. For example, using TightM with CPLEX features enabled (the fifth column in table 4), incumbents within 1% of optimal solutions are found in 1 minute for all 10 instances. Moreover, after 10 minutes TightM with CPLEX features enabled found incumbents within 0.01% of the optimal solutions for 4 instances, yet 3 of them took more than 12 hours to eventually be solved to optimality.

In the second set of experiments we aim to provide practical advice for solving large scale problems. All CPLEX advanced features are enabled. We considered 10 instances of reinsurance problems with 10 000 scenarios and 25 risk contracts. A 99.5%-VaR minimization problem contains 50 tail scenarios, that is, $m^* = 50$. The number of scenarios and number of tail scenarios of this size are similar to problems considered in practice. In this case we set a time limit of 12 hours to replicate real-life situations where an over-night computational budget is allowed. The results are summarized in the sixth to the ninth columns

Table 3. CPU time (in seconds, including big-M calculation time) and optimality gap for solving 99.5%-VaR minimization problems with 25 risk contracts.

	5000 scenarios and 3-hours limit				10 000 scenarios and 12-hours limit			
	NaturalM		TightM		NaturalM		TightM	
Instance ID	CPU time	Opt. gap	CPU time	Opt. gap	CPU time	Opt. gap	CPU time	Opt. gap
1	2376.8	(–)	149.9	(–)	(–)	9.75%	(–)	2.10%
2	(–)	6.93%	(25 071.3)	2.04%	(–)	14.44%	(–)	7.97%
3	(–)	10.88%	(134 222.2)	6.18%	691.3	(–)	97.0	(–)
4	53.6	(–)	29.9	(–)	(–)	11.21%	10 128.5	(–)
5	111.8	(–)	41.0	(–)	13 568.1	(–)	757.1	(–)
6	1162.2	(–)	119.7	(–)	(–)	9.52%	34 225.9	(–)
7	3993.1	(–)	281.8	(–)	(–)	10.12%	(–)	1.95%
8	609.6	(–)	182.0	(–)	(–)	2.77%	16 144.2	(–)
9	(–)	8.68%	(17 423.2)	1.66%	(1807.9)	15.95%	(–)	8.10%
10	(–)	3.35%	1859.8	(–)	(–)	17.63%	(–)	5.95%

Notes: CPLEX features enabled. Number of scenarios and time limit are as specified in table. CPU time in a parenthesis denotes the eventual solve time of an instance.

Table 4. Percentage gaps between best incumbent and optimal solution for solving 99.5%-VaR minimization problems with 5000 scenarios and 25 risk contracts.

	Incumbent gap after 1 minute				Incumbent gap after 10 minutes			
	Features disabled		Features enabled		Features disabled		Features enabled	
Instance ID	NaturalM	TightM	NaturalM	TightM	NaturalM	TightM	NaturalM	TightM
1	0.28%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	Solved
2	1.76%	1.10%	0.70%	0.78%	0.21%	0.00%	0.02%	0.01%
3	1.67%	1.81%	0.44%	0.65%	1.67%	0.00%	0.00%	0.00%
4	0.00%	Solved	Solved	Solved	0.00%	Solved	Solved	Solved
5	8.65%	0.00%	Solved	Solved	0.00%	Solved	Solved	Solved
6	2.61%	0.63%	0.00%	0.00%	0.38%	0.00%	0.00%	Solved
7	1.07%	0.70%	0.39%	0.21%	0.00%	0.00%	0.00%	Solved
8	0.94%	0.76%	0.00%	0.75%	0.00%	Solved	Solved	Solved
9	0.37%	0.37%	0.37%	0.63%	0.37%	0.37%	0.00%	0.00%
10	0.89%	0.44%	0.79%	0.00%	0.44%	0.00%	0.44%	0.00%

of table 3. Out of 10 large scale problems, TightM solved 5 instances within the time limit and had a final optimality gap less than 3% for 7 instances. NaturalM, however, solved only 2 instances within the time limit and had 3 instances whose optimality gaps were less than 3%. For these large instances TightM is 7–18 times faster than NaturalM when an instance is solved within the time limit. When an instance is halted, TightM always has a smaller optimality gap than NaturalM.

6. Conclusion

We examined several algorithms to solve VaR-minimization problems. Our experiments suggest that calculating tight big-Ms improves overall performance. When CPLEX's advanced features are disabled, variations of the specialized decomposition algorithm perform comparably to the MIP formulation with tight big-Ms. We hope to see significant improvements in computational performance if we were able to enable some of the CPLEX features for the decomposition algorithm. The efficacy of these methods when VaR-constrained problems remains to be investigated. At the current stage, we recommend TightM as a practical method to solve VaR problems.

Acknowledgments

The authors thank Dr James Luedtke and Dr Simge Küçükyavuz for offering insights into this work. Mingbin Feng and Andreas Wächter were supported partially by the NSF grant DMS-1216920.

References

Artzner, P., Delbaen, F., Eber, J.M. and Heath, D., Coherent measures of risk. *Math. Financ.*, 1999, **9**, 203–228.

Basel Committee on Banking Supervision, Basel II: International convergence of capital measurement and capital standards. A revised framework. Bank for International Settlements, 2004.

Basel Committee on Banking Supervision, Basel III: A global regulatory framework for more resilient banks and banking systems. Bank for International Settlements, 2010.

Benati, S. and Rizzi, R., A mixed integer linear programming formulation of the optimal mean/value-at-risk portfolio problem. *Eur. J. Oper. Res.*, 2007, **176**, 423–434.

Gilli, M. and Këllezi, E., A global optimization heuristic for portfolio choice with VaR and expected shortfall. In *Computational Methods in Decision-making, Economics and Finance, Applied Optimization Series*, edited by E.J. Kontoghiorghes, B. Rustem, and S. Siokos, pp. 167–183, 2002 (Springer: New York).

Günlük, O. and Pochet, Y., Mixing mixed-integer inequalities. *Math. Program.*, 2001, **90**, 429–457.

Hong, L.J., Hu, Z. and Zhang, L., Conditional value-at-risk approximation to value-at-risk constrained programs: A remedy via Monte Carlo. *INFORMS J. Comput.*, 2014, **26**, 385–400.

Hong, L.J., and Liu, G., Simulating sensitivities of conditional value at risk. *Management Science*, 2009, **55**, 281–293.

IBM ILOG, IBM ILOG CPLEX Optimization Studio. Getting Started with CPLEX., 2013, Version 12 Release 5.

Jorion, P., *Value at Risk: The New Benchmark for Controlling Market Risk*, 1997 (McGraw-Hill: New York).

Kou, S. and Peng, X., Comments on the consultative document 'Fundamental Review of the Trading Book' Released by Bank for International Settlement on May 3rd, 2012.

Kou, S., Peng, X. and Heyde, C.C., External risk measures and basel accords. *Math. Oper. Res.*, 2013, **38**, 393–417.

Krokhmal, P., Palmquist, J. and Uryasev, S., Portfolio optimization with conditional value-at-risk objective and constraints. *J. Risk*, 2002, **4**, 43–68.

Larsen, N., Mausser, H. and Uryasev, S., Algorithms for Optimization of Value-at-Risk. In *Financial Engineering, e-Commerce and Supply Chain, Applied Optimization Series*, edited by P. Pardalos, and V. Tsitsiringos, pp. 19–46, 2002 (Springer: New York).

Lim, C., Sherali, H.D. and Uryasev, S., Portfolio optimization by minimizing conditional value-at-risk via nondifferentiable optimization. *Comput. Optim. Appl.*, 2010, **46**, 391–415.

Lin, C.-C., Comments on 'A mixed integer linear programming formulation of the optimal mean/value-at-risk portfolio problem'. *Eur. J. Oper. Res.*, 2009, **194**, 339–341.

Luedtke, J., A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Math. Program.*, 2013, **146**, 219–244.

Ogryczak , W. and Śliwiński, T., On solving the dual for portfolio selection by optimizing conditional value at risk. *Comput. Optim. Appl.*, 2011, **50**, 591–595.

Pritsker, M., Evaluating value at risk methodologies: Accuracy versus computational time. *J. Financ. Serv. Res.*, 1997, **12**, 201–242.

Qiu, F., Ahmed, S., Dey, S.S. and Wolsey, L.A., Covering linear programming with violations. Working paper, 2012.

Rockafellar, R.T., and Uryasev, S., Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 2002, **26**, 1443–1471.

Song, Y., Luedtke, J. and Küçükyavuz, S., Chance-constrained binary packing problems. Working paper, 2012.

Appendix 1. Proof for Proposition 3.2

Proof Let i be a pivot scenario. Inequality (6a) is backed by (2b), (2c) and the definition of RELs. Since Equation (2b) implies $\{i : L_i.x > z\} \subseteq \{i : b_i = 1\}$ and Equation (2c) implies $|\{i : b_i = 1\}| \leq m^*$, thus $|\{i : L_i.x > z\}| \leq m^*$. By definition of RELs and the ordering (5) we know that $\{k : L_k.x \geq L_i.x - d_{\sigma_{m^*+1}(i)}(i)\} \subseteq \{\sigma_1(i), \ldots, \sigma_{m^*+1}(i)\}$. Thus $|\{k : L_k.x \geq L_i.x - d_{\sigma_{m^*+1}(i)}(i)\}| \geq m^* + 1$. Assume to the contrary that Equation (6a) is violated by any solution $(\hat{x}, \hat{b}, \hat{z})$, that is, $L_i.\hat{x} - d_{\sigma_{m^*+1}(i)}(i) > \hat{z}$. Then we have $m^* + 1 \leq |\{k : L_k.x \geq L_i.x - d_{\sigma_{m^*+1}(i)}(i)\}| \leq |\{i : L_i.x > z\}| \leq m^*$, which is a contradiction. Therefore Equation (6a) is valid for all feasible solution to Equation (2).

The implication constraint (6b) results from the definition of REL and the implication constraint (2b). By Equation (2b) we know that $b_{\sigma_k} = 0 \Rightarrow L_{\sigma_k}x \leq z$ for all $k = 1, \ldots, m$. So $b_{\sigma_k} = 0$ implies that $L_i.x - z \leq L_i.x - L_{\sigma_k}.x \leq \max\{(L_i. - L_{\sigma_k}.)x | x \in \mathcal{X}\}$, which validates the implication constraint (6b).

Constraints (6c) is a consequence of (6a), (6b) and the ordering (5). This completes the proof for Proposition 3.2. ■

Appendix 2. Specialized decomposition algorithm for VaR problems

Algorithm 1: Branch-and-cut decomposition algorithm for solving VaR problems

$t \leftarrow 0, N_0(0) \leftarrow \emptyset, N_1(0) \leftarrow \emptyset, C \leftarrow \mathbb{R}^{m+n+1}$, OPEN $\leftarrow \{0\}, U \leftarrow \infty$;
while *OPEN* $\neq \emptyset$ **do**
  Step 1: Choose $l \in$ OPEN and let OPEN $\leftarrow$ OPEN $\setminus \{l\}$;
  Step 2: Process node l;
  repeat
    Solve (9);
    if *(9) is infeasible* **then**
      CUTFOUND $\leftarrow$ FALSE;
    else
      Let $(\hat{x}, \hat{b}, \hat{z})$ be an optimal solution to (9);
      lb $\leftarrow MP(N_0(l), N_1(l), C)$;
      if $\hat{b} \in \{0,1\}^m$ **then**
        CUTFOUND $\leftarrow$ SepCuts$(\hat{x}, \hat{b}, \hat{z}, C)$;
        if CUTFOUND = FALSE **then** $U \leftarrow$ lb;
      else
        CUTFOUND $\leftarrow$ FALSE;
        Optional: CUTFOUND $\leftarrow$ SepCuts$(\hat{x}, \hat{b}, \hat{z}, C)$;
      end
    end
  until CUTFOUND $\neq$ TRUE or lb $\geq U$;
  Step 3: Branch if necessary;
  if lb $< U$ **then**
    Choose $i \in \{1, \ldots, m\}$ such that $\hat{b}_i \in (0,1)$;
    $N_0(t+1) \leftarrow N_0(l) \bigcup \{i\}, N_1(t+1) \leftarrow N_1(l)$;
    $N_0(t+2) \leftarrow N_0(l), N_1(t+2) \leftarrow N_1(l) \bigcup \{i\}$;
    $t \leftarrow t+2$;
    OPEN $\leftarrow$ OPEN$\bigcup \{t+1, t+2\}$;
  end
end

Algorithm 2: SepCuts$(\hat{x}, \hat{b}, \hat{z}, C)$

Data: $\hat{x}, \hat{b}, \hat{z}, C$
Result: If valid inequalities for feasible region of (9) are found that are violated by $(\hat{x}, \hat{b})$, add them to the description of C and return TRUE, else return FALSE

CUTFOUND $\leftarrow$ FALSE;
for $i \in \{1, \ldots, m\}$ **do**
  if $\hat{b}_i < 1$ and $L_i.\hat{x} > \hat{z}$ **then**
    Calculate $d_j(i)$ as defined in (4) for all $j = 1, \ldots, m$;
    Sort $d_j(i), j = 1, \ldots, m$ according to (5) to obtain permutation $\sigma(i)$;
    Set $j^* = 1$, $\rho_{j^*} = \sigma_1$, $R^* = \{\rho_{j^*}\}$;
    for $k = 2, \ldots, m^*$ **do**
      If $\hat{b}_{\sigma_k} < \hat{b}_{\rho_{j^*}}$ then set $j^* = j^* + 1$, $\rho_{j^*} = \sigma_k$, $R^* = R^* \cup \rho_{j^*}$;
    end
    If inequality of form (12) based on R^* is violated by $(\hat{x}, \hat{b}, \hat{z})$, add a non-empty set of violated inequalities to the description of C;
    CUTFOUND $\leftarrow$ TRUE;
    break;
  end
end
return CUTFOUND and updated C;