Optimization of Value-at-Risk: Computational Aspects of MIP Formulations

Konstantin Pavlikov

Department of Industrial and Systems Engineering
University of Florida

November 9, 2015

Introduction

- Value-at-Risk is a standard tool in financial industry to measure and control various types of risks, compute regulatory capital.
- It answers the question "What is the minimum threshold value that my loss does not exceed with probability at least α?"
  (Thus, my loss exceeds the threshold with probability at most $1 - \alpha$)

Consider Value-at-Risk minimization problem, which can be motivated as follows: given that my loss exceeds the threshold value (VaR) with 5% probability, I prefer it (threshold) to be as small as possible!

Other applications:

- Portfolio allocation: Gaivoronski and Pflug (2005), Feng, Wachter and Staum (2015)
- Facility location: Daskin, Hesse and Revelle (1997); Chen, Daskin, Shen and Uryasev (2006)
- Support vector machines: Tsyurmasto, Zabarankin and Uryasev (2014)
- Statistics: Least median of squares regression, Rousseeuw (1984)

VaR Definition

Definition:

$$\mathrm{VaR}_\alpha(\mathcal{L}) = \inf\{l \in \mathbb{R} : \mathbb{P}(\mathcal{L} > l) \leq 1 - \alpha\}, \qquad \alpha \in (0, 1]$$

Example: Loss is distributed uniformly

L_j	-7	-3	-1	2	3
p_j	0.2	0.2	0.2	0.2	0.2
$\mathbb{P}(\mathcal{L} > L_j)$	0.8	0.6	0.4	0.2	0

$$\mathrm{VaR}_{(0.8,1]}(\mathcal{L}) = 3 \qquad \mathrm{VaR}_{(0.6,0.8]}(\mathcal{L}) = 2$$

Key Properties of VaR:

- $$\mathrm{VaR}_\alpha(\mathcal{L}) = L_j \quad \text{for some } j\,.$$
- Translation Invariance

$$\mathrm{VaR}_\alpha(C + \mathcal{L}) = C + \mathrm{VaR}_\alpha(\mathcal{L})$$

- Monotonicity

$$\begin{pmatrix} L_1 \\ L_2 \\ \dots \\ L_Q \end{pmatrix} \leq \begin{pmatrix} L_1^{'} \\ L_2^{'} \\ \dots \\ L_Q^{'} \end{pmatrix} \implies \mathrm{VaR}_\alpha \begin{pmatrix} L_1 \\ L_2 \\ \dots \\ L_Q \end{pmatrix} \leq \mathrm{VaR}_\alpha \begin{pmatrix} L_1^{'} \\ L_2^{'} \\ \dots \\ L_Q^{'} \end{pmatrix}.$$

- Inversion

$$\mathrm{VaR}_\alpha(-\mathcal{L}) = -\mathrm{VaR}_{1-\alpha}(\mathcal{L})$$

VaR and CVaR Illustration

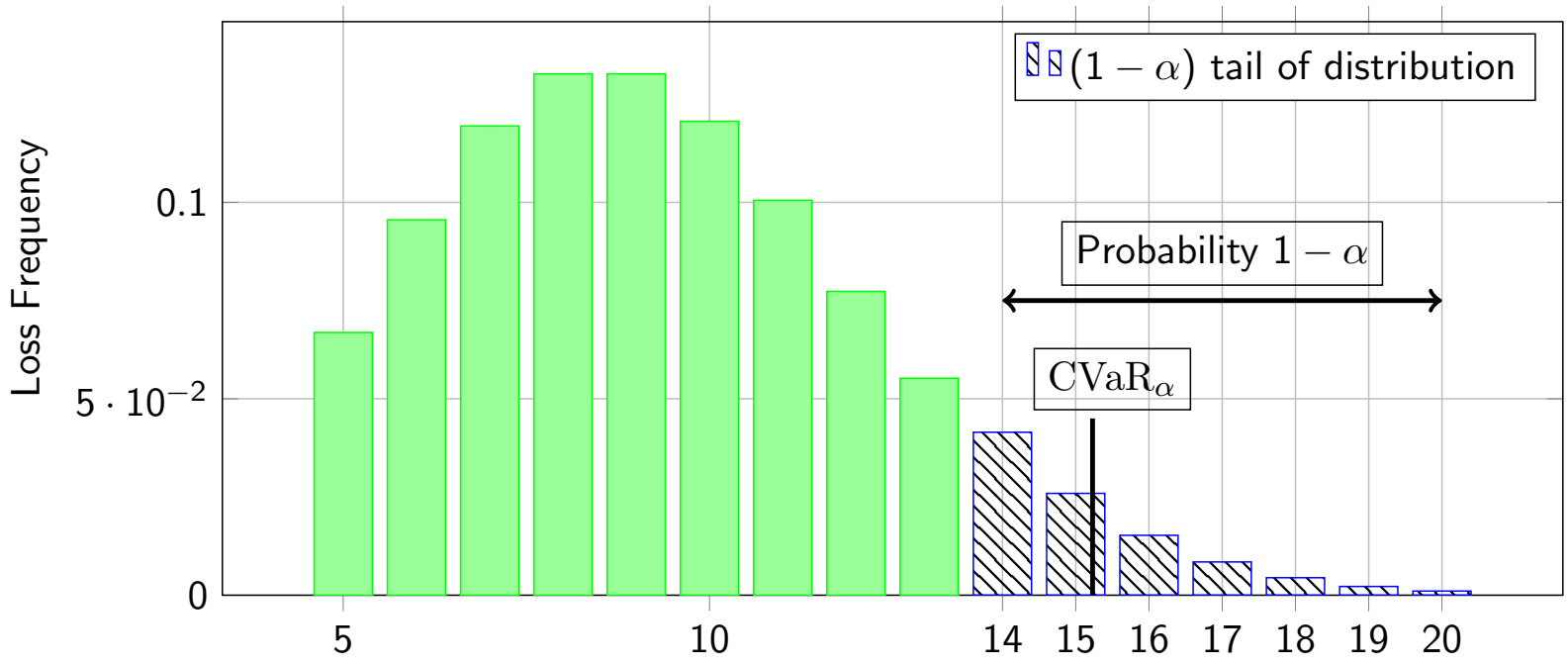


Figure: $\mathrm{VaR}_\alpha(\mathcal{L}) = 14, \quad \mathrm{CVaR}_\alpha(\mathcal{L}) \approx 15.2$

Settings

- X is a convex bounded or a discrete bounded set of possible decisions, $\mathbf{x} \in X$
- $\mathcal{L}(\mathbf{x})$ is a r.v. of losses with outcomes $\{L_1(\mathbf{x}), \ldots, L_Q(\mathbf{x})\}$ with probabilities $\{p_1, \ldots, p_Q\}$

$L_j(\mathbf{x})$ is a linear function of $\mathbf{x}$ and scenario information S_j:

$$L_j(\mathbf{x}) = f(S_j, \mathbf{x})$$

$$\underline{L}_j = \min_{\mathbf{x}} L_j(\mathbf{x}) < \overline{L}_j = \max_{\mathbf{x}} L_j(\mathbf{x})$$

Minimum VaR problem :

$$\min_{\mathbf{x} \in X} \ \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) \iff \min_{\mathbf{x} \in X} \ l \qquad (1)$$

subject to

$$\mathbb{P}(\mathcal{L}(\mathbf{x}) > l) \leq 1 - \alpha \,.$$

Nonconvexity

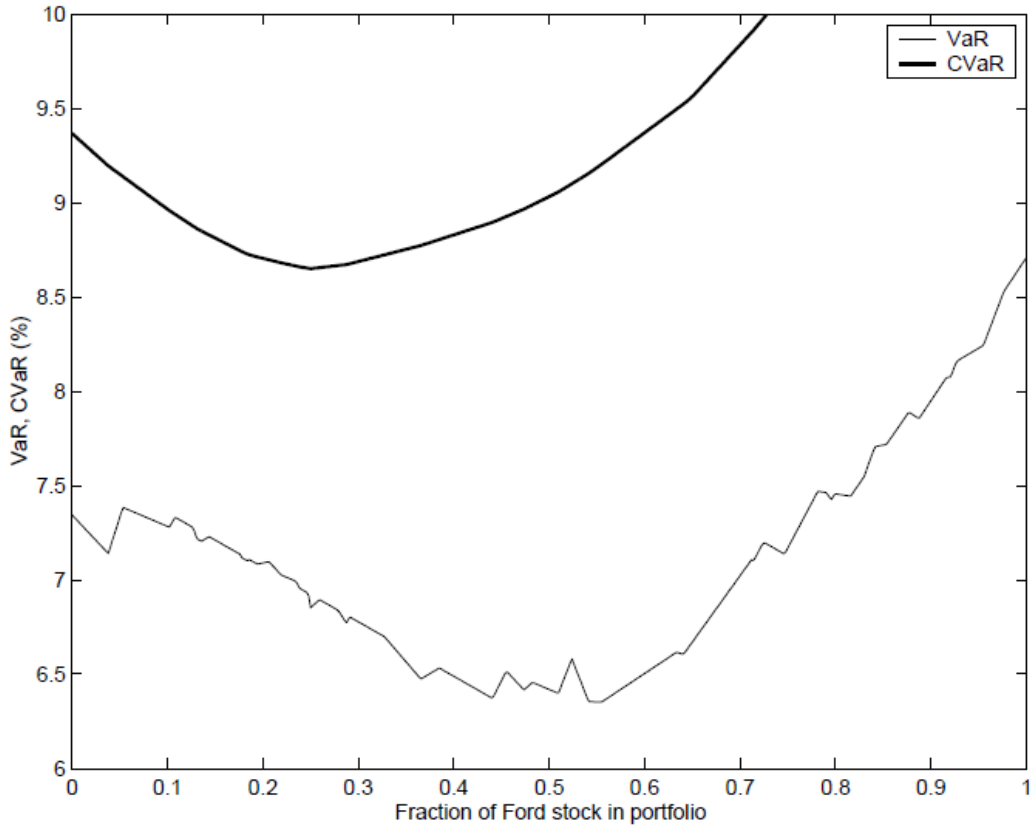


Figure: VaR nonconvexity example from Gaivoronski and Pflug (2005)

Solution approaches:

- Heuristics. Larsen et al. (2002)
- MIP optimization with big *M*s. Feng, Wachter and Staum (2015)
- Chance-constrained optimization. Luedtke (2013), Feng, Wachter and Staum (2015)
- MIP optimization with SOS1 constraints. Bertsimas and Mazumder (2014)

MIP Optimization with Big Ms

In order to determine the $\mathbb{P}(\mathcal{L}(\mathbf{x}) > l)$, we also introduce a set of indicator variables $z_j \in \{0,1\}\,,\ j = 1,\ldots,Q$, such that

$$z_j = 1 \quad \Longrightarrow \quad L_j(\mathbf{x}) > l\,, \tag{2}$$
$$z_j = 0 \quad \Longrightarrow \quad L_j(\mathbf{x}) \leq l\,. \tag{3}$$

(2) is equivalent to $z_j \geq \dfrac{L_j(\mathbf{x}) - l}{M_j}$, which can only be correctly defined if M_j is "big" enough:

$$L_j(\mathbf{x}) - l \leq M_j \qquad \forall \mathbf{x} \in X\,. \tag{4}$$

MIP formulation:

$$\begin{aligned}
&\min \quad l \\
&\text{subject to} \\
&z_j \geq \frac{L_j(\mathbf{x}) - l}{M_j}\,, && j = 1,\ldots,Q\,, \\
&\sum_{j=1}^{Q} p_j z_j \leq 1 - \alpha\,, \\
&z_j \in \{0,1\}\,, && j = 1,\ldots,Q\,,
\end{aligned}$$

Big *M*s, Feng, Wachter and Staum (2015)

Recently, efficient $\widetilde{M}_j$ have beed proposed

$$d_t(j) = \max_{\mathbf{x}\in X} \; L_j(\mathbf{x}) - L_t(\mathbf{x})\,, \qquad t = 1,\ldots,Q\,. \tag{5}$$

$$\widetilde{M}_j = \mathrm{VaR}_{1-\alpha}\left(d_1(j),\ldots,d_Q(j)\right)\,, \qquad j = 1,\ldots,Q\,. \tag{6}$$

Proof.

$$\mathrm{VaR}_{1-\alpha}\left(L_j(\mathbf{x}) - \mathcal{L}(\mathbf{x})\right) = L_j(\mathbf{x}) + \mathrm{VaR}_{1-\alpha}\left(-\mathcal{L}(\mathbf{x})\right) =$$
$$L_j(\mathbf{x}) - \mathrm{VaR}_{\alpha}\left(\mathcal{L}(\mathbf{x})\right) = L_j(\mathbf{x}) - I\,.$$

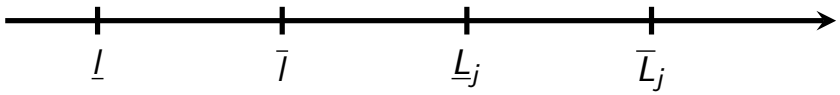
$$\mathrm{VaR}_{1-\alpha}\left(L_j(\mathbf{x}) - \mathcal{L}(\mathbf{x})\right) =$$
$$\mathrm{VaR}_{1-\alpha}\begin{pmatrix} L_j(\mathbf{x}) - L_1(\mathbf{x}) \\ L_j(\mathbf{x}) - L_2(\mathbf{x}) \\ \ldots \\ L_j(\mathbf{x}) - L_Q(\mathbf{x}) \end{pmatrix} \le \mathrm{VaR}_{1-\alpha}\begin{pmatrix} \max\limits_{\mathbf{x}}\left(L_j(\mathbf{x}) - L_1(\mathbf{x})\right) \\ \max\limits_{\mathbf{x}}\left(L_j(\mathbf{x}) - L_2(\mathbf{x})\right) \\ \ldots \\ \max\limits_{\mathbf{x}}\left(L_j(\mathbf{x}) - L_Q(\mathbf{x})\right) \end{pmatrix} = \widetilde{M}_j\,.$$

Remark. We can safely remove scenarios with $\widetilde{M}_j \le 0$ *from formulation.* $\mathcal{M}^+$ is the set of scenarios with positive $\widetilde{M}_j$, $\mathcal{M}^-$ – with nonpositive.

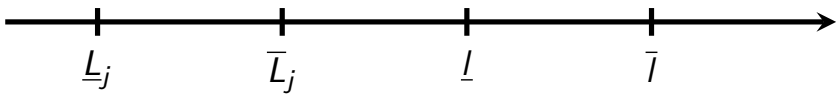
Scenario Classification

$$\underline{l} \le l^* \le \bar{l}\,, \qquad \underline{l} < \bar{l}\,.$$

- $\bar{l} < \underline{L}_j \implies z_j = 1\,.$ Let $\mathcal{Q}_1 = \{j \in \{1,\dots,Q\} \mid \bar{l} < \underline{L}_j\}$



- $\bar{L}_j \le \underline{l} \implies z_j = 0\,.$ Let $\mathcal{Q}_2 = \{j \in \{1,\dots,Q\} \mid \bar{L}_j \le \underline{l}\}\,.$



- $\bar{l} \ge \underline{L}_j$ and $\bar{L}_j > \underline{l}\,,$ $\mathcal{Q}_3 = \{1,\dots,Q\}\backslash(\mathcal{Q}_1 \cup \mathcal{Q}_2)$

$$\underline{L}_j - \bar{l} \le L_j(\mathbf{x}) - l \le \bar{L}_j - \underline{l}\,.$$

which implies that the lower bound $\underline{l}$ defines the $M_j = \bar{L}_j - \underline{l}$ and that the minimum of M_j and $\widetilde{M}_j$ can be used in the formulation that uses classification $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$

$$\min \quad l \tag{7}$$

subject to

$$z_j \ge \frac{L_j(\mathbf{x}) - l}{\min(\widetilde{M}_j, \bar{L}_j - \underline{l})}\,, \qquad j \in \mathcal{Q}_3 \cap \mathcal{M}^+\,, \tag{8}$$

$$\underline{l} \le l \le \bar{l}\,, \tag{9}$$

$$\sum_{j \in \mathcal{Q}_3 \cap \mathcal{M}^+} p_j z_j \le 1 - \alpha - \sum_{j \in \mathcal{Q}_1} p_j\,, \tag{10}$$

$$z_j \in \{0,1\}\,, \qquad j \in \mathcal{Q}_3 \cap \mathcal{M}^+\,. \tag{11}$$

Lower Bound Lifting

Let $\underline{l}^0$ be the initial lower bound on the optimal solution. Next bound is defined as:

$$
\begin{aligned}
&\underline{l}^1 = \min \quad l \\
&\text{subject to} \\
&z_j \geq \frac{L_j(\mathbf{x}) - l}{\min(\widetilde{M}_j, \overline{L}_j - \underline{l}^0)}\,, && j \in \mathcal{Q}_3 \cap \mathcal{M}^+\,, \\
&\underline{l}^0 \leq l \leq \bar{l}\,, \\
&\sum_{j \in \mathcal{Q}_3 \cap \mathcal{M}^+} p_j z_j \leq 1 - \alpha - \sum_{j \in \mathcal{Q}_1} p_j\,, \\
&z_j \in [0,1]\,, && j \in \mathcal{Q}_3 \cap \mathcal{M}^+\,.
\end{aligned}
$$

Clearly,

$$\underline{l}^1 \geq \underline{l}^0$$

and we continue the iterative proccess until $\underline{l}^{k+1} - \underline{l}^k < \epsilon$

Lower Bound Lifting

$Q =$ $n =$ $\alpha =$	200 10 150/200	250 10 175/250	250 10 200/250	300 10 250/300	350 10 275/350	350 10 300/350
$\bar{I}$	4.6126	3.1114	6.3921	7.3258	6.1729	8.7418
$\underline{I}^0$	−14.572	−16.667	−12.270	−11.136	−14.925	−10.776
$\underline{I}^1$	−6.1388	−8.3930	−3.8045	−2.6224	−5.3480	−1.5655
$\underline{I}^2$	−6.1156	−8.3542	−3.7707	−2.5877	−5.3179	−1.5260
$\underline{I}^3$	−6.1148	−8.3529	−3.7691	−2.5864	−5.3167	−1.5246
$\underline{I}^4$	–	–	–	–	−5.3166	−1.5245
$\lvert\mathcal{Q}_1\rvert$	4	8	5	5	5	2
$\lvert\mathcal{Q}_2^k\rvert$	2	0	2	2	2	3
$\lvert\mathcal{M}^-\rvert$	2	1	4	4	3	6
$\lvert\mathcal{Q}_2^k \cup \mathcal{M}^-\rvert$	3	1	4	4	4	6
$\#j : 0 < \bar{L}_j - \underline{I}^k < \widetilde{M}_j$	14	13	26	23	27	28
CPU total (sec.)	2.49	4.16	3.25	4.30	6.44	4.43

Two Stage Solution Procedure

- It is important to apply a good heuristic to obtain an upper bound, $\bar{l}$, to potentially increase the cardinality of set $\mathcal{Q}_1$
- It is important to obtain a tight lower bound, $\underline{l}^k$, to potentially increase the cardinality of set $\mathcal{Q}_2$
- Tight lower bound defines tight M_j values

Is there any other tool to tighten the bounds?

Two Stage Solution Procedure

- It is important to apply a good heuristic to obtain an upper bound, $\bar{l}$, to potentially increase the cardinality of set $\mathcal{Q}_1$
- It is important to obtain a tight lower bound, $\underline{l}^k$, to potentially increase the cardinality of set $\mathcal{Q}_2$
- Tight lower bound defines tight M_j values

Is there any other tool to tighten the bounds?

Yes, the standard MIP solver!

Two Stage Solution Procedure

- It is important to apply a good heuristic to obtain an upper bound, $\bar{l}$, to potentially increase the cardinality of set $\mathcal{Q}_1$
- It is important to obtain a tight lower bound, $\underline{l}^k$, to potentially increase the cardinality of set $\mathcal{Q}_2$
- Tight lower bound defines tight M_j values

Is there any other tool to tighten the bounds?

Yes, the standard MIP solver!

- The branch-and-bound algorithm behind the solver continuously updates upper and lower bounds until they coinside
- Let solver run for a prespecified number of BnB nodes N, then stop the solver and record:
  - new lower bound
  - new upper bound
  - new best solution
- Redefine $\mathcal{Q}_1$, $\mathcal{Q}_2$, big Ms and restrart the solver to be run to optimality

Two Stage Solution Procedure

The solution procedure is outlined as follows:

- Use the best possible heuristic to obtain $\bar{I}$ and the corresponding feasible solution $\mathbf{x}_0$; define $\mathcal{Q}_1$
- Find big $\widetilde{M}_j$ according to Feng et al. (2015)
- Find the initial lower bound $\underline{I}^0$ and run the iterative lifting procedure to find best possible bound $\underline{I}^k$; define $\mathcal{Q}_2$

- Stage 1: Run the formulation with $\mathcal{Q}_1$, $\mathcal{Q}_3$ using $\mathbf{x}_0$ as the "warm" start for the solver; stop it when N branch-and-bound nodes have been solved.
  - new upper bound $\bar{I} \implies$ new $\mathcal{Q}_1$
  - new lower bound $\underline{I}^{k+1} \implies$ new $\mathcal{Q}_2$
  - new best feasible solution $\mathbf{x}_1$

- Stage 2: Run the tightened formulation with $\mathbf{x}_1$ as the new "warm" start, to optimality

First Stage Results

$Q =$	200	250	250	300	350	350
$n =$	10	10	10	10	10	10
$\alpha =$	150/200	175/250	200/250	250/300	275/350	300/350
$\bar{l}$	4.6126	3.1114	6.3921	7.3258	6.1729	8.7418
$\underline{l}^0$	−14.572	−16.667	−12.270	−11.136	−14.925	−10.776
$\underline{l}^1$	−6.1388	−8.3930	−3.8045	−2.6224	−5.3480	−1.5655
$\underline{l}^2$	−6.1156	−8.3542	−3.7707	−2.5877	−5.3179	−1.5260
$\underline{l}^3$	−6.1148	−8.3529	−3.7691	−2.5864	−5.3167	−1.5246
$\underline{l}^4$	–	–	–	–	−5.3166	−1.5245
$\lvert\mathcal{Q}_1\rvert$	4	8	5	5	5	2
$\lvert\mathcal{Q}_2^4\rvert$	2	0	2	2	2	3
$\lvert\mathcal{M}^-\rvert$	2	1	4	4	3	6
$\lvert\mathcal{Q}_2^4 \cup \mathcal{M}^-\rvert$	3	1	4	4	4	6
$\#j : 0 < \bar{L}_j - \underline{l}^4 < \widetilde{M}_j$	**14**	**13**	**26**	**23**	**27**	**28**
CPU total (sec.)	2.49	4.16	3.25	4.30	6.44	4.43
$\bar{l}^1$	3.2047	2.6128	5.5949	6.5253	4.8675	8.0982
$\underline{l}^5$	−1.0191	−3.8803	0.6905	2.0652	−0.9117	2.3489
$\lvert\mathcal{Q}_1^1\rvert$	6	8	6	5	6	2
$\lvert\mathcal{Q}_2^5 \cup \mathcal{M}^-\rvert$	3	2	5	6	4	9
$\#j : 0 < \bar{L}_j - \underline{l}^5 < \widetilde{M}_j$	**79**	**82**	**103**	**104**	**113**	**105**
CPU (sec.)	28.40	44.25	36.17	40.65	55.72	55.90

Table: First stage results after $N = 100,000$ BnB nodes.

Overall Computational Results

α	N	Q	Obj	Benchmark	Two stage
150/200	50,000	200	2.3601	894.09	129.01
175/250	50,000	250	1.4178	–	2,959.96
200/250	50,000	250	4.0423	6,376.12	117.14
250/300	100,000	300	4.9235	1,734.79	424.75
275/350	100,000	350	3.3188	–	5,278.72
300/350	100,000	350	5.8179	5,501.75	727.58
350/400	200,000	400	6.4551	9,063.21	1,075.8
425/475	200,000	475	7.4338	–	3,432.05

Table: CPU time in seconds, benchmark vs. two-stage solution approach. "–" denotes instances with out of memory error

Conclusion

- The importance of bounds on optimal VaR has been demonstrated
- With minimum coding and just using the solver as the only tool, it is possible to cut the solution time by up to 80%
- There is a potential for a specialized BnB algorithm for such type of problems that will be updating the LP matrix on fly as the bounds change – something that can not be currently done using the functionality of commercial solvers

Setting the Tight M_j

$$L_j(\mathbf{x}) - \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) \leq \mathbf{M}_j \qquad \forall \mathbf{x} \in X \implies$$

$$\begin{aligned}
\mathbf{M}_j &= \max_{\mathbf{x}\in X} \; L_j(\mathbf{x}) - \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) = \\
&- \min_{\mathbf{x}\in X} \; -L_j(\mathbf{x}) + \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) = \\
&- \min_{\mathbf{x}\in X} \; \mathrm{VaR}_\alpha\left(\mathcal{L}(\mathbf{x}) - L_j(\mathbf{x})\right) .
\end{aligned}$$

Let $\underline{l}_j$ and $\bar{l}_j$ denote a lower and an upper bounds to the above VaR minimization problem:

$$\begin{aligned}
\underline{l}_j \leq \min_{\mathbf{x}\in X} \; \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_j(\mathbf{x})) \leq \bar{l}_j \, , \\
-\bar{l}_j \leq -\min_{\mathbf{x}\in X} \; \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_j(\mathbf{x})) \leq -\underline{l}_j \, ,
\end{aligned}$$

therefore,

$$\mathbf{M}_j \leq -\underline{l}_j \;\; \text{(iteratively lifted)} \, .$$

Setting the Tight M_j

$$\min_{\mathbf{x}} \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}))$$

$$\mathbf{M}_1 = -\min_{\mathbf{x}} \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_1(\mathbf{x})) \quad \ldots \quad \mathbf{M}_Q = -\min_{\mathbf{x}} \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_Q(\mathbf{x}))$$

Figure: A scheme for computation of big M values for a general VaR optimization problem.

$$\min_{\mathbf{x}} \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_j(\mathbf{x}))$$

$$\mathbf{M}_1^j = -\min_{\mathbf{x}} \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_1(\mathbf{x})) \quad \ldots \quad \mathbf{M}_Q^j = -\min_{\mathbf{x}} \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x}) - L_Q(\mathbf{x}))$$

Figure: A scheme for computation of big M values for a subproblem of the VaR optimization problem. Thus $\mathbf{M}_t^j = \mathbf{M}_t$.

Restricting the Feasible Space X

$$L_j(\mathbf{x}) - \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) \leq \mathbf{M}_j \qquad \forall \mathbf{x} \in X$$

However, we already know that $\underline{l} \leq \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) \leq \bar{l}$, therefore

$$X' \subset X : \quad \underline{l} \leq \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) \leq \bar{l}$$

This restricted feasible set can now be used to redefine other constants

- $$\underline{L}'_j = \min_{\mathbf{x} \in X'} L_j(\mathbf{x}) < \bar{L}'_j = \max_{\mathbf{x} \in X'} L_j(\mathbf{x})$$
- $$d'_t(j) = \max_{\mathbf{x} \in X'} \; L_j(\mathbf{x}) - L_t(\mathbf{x}) \,, \qquad t = 1, \ldots, Q \,.$$
$$\widetilde{M}'_j = \mathrm{VaR}_{1-\alpha} \left(d'_1(j), \ldots, d'_Q(j) \right) \,, \qquad j = 1, \ldots, Q \,.$$

Connection to General Chance-Constrained Problems

$$\min_{\mathbf{x}\in X} \quad f(\mathbf{x})$$

subject to

$$\mathbb{P}(\mathcal{L}(\mathbf{x}) \leq 0) \geq \alpha \iff \mathbb{P}(\mathcal{L}(\mathbf{x}) > 0) \leq 1-\alpha \iff \mathrm{VaR}_\alpha(\mathcal{L}(\mathbf{x})) \leq 0\,.$$

Reformulated with big Ms:

$$\begin{aligned}
&\min_{\mathbf{x}\in X} \quad f(\mathbf{x}) \\
&\text{subject to} \\
&l \leq 0\,, \\
&z_j \geq \frac{L_j(\mathbf{x}) - l}{M_j}\,, && j = 1,\ldots,Q\,, \\
&\sum_{j=1}^{Q} p_j z_j \leq 1-\alpha\,, \\
&z_j \in \{0,1\}\,, && j = 1,\ldots,Q\,.
\end{aligned}$$

An upper bound is obtained by solving the following convex problem:

$$\begin{aligned}
&\overline{f} = \min_{\mathbf{x}\in X} \quad f(\mathbf{x}) \\
&\text{subject to} \\
&\mathrm{CVaR}_\alpha(\mathcal{L}(\mathbf{x})) \leq 0\,.
\end{aligned}$$

Connection to General Chance-Constrained Problems

$$X' \subset X : \quad f(\mathbf{x}) \leq \overline{f}$$

Big Ms can be calculated based on the restricted feasiblie set X':

$$M_j := \max_{\mathbf{x} \in X'} \; \mathcal{L}(\mathbf{x}) - l\,, \qquad\qquad j = 1, \ldots, Q\,.$$

The outline of a special branch-and-bound algorithm for this problem:

- Solve the problem via the CVaR restriction to obtain X'
- Find $M_j\,,\ j = 1, \ldots, Q$ (possibly, approximately)
- Once branch and bound algorithm finds a new upper bound $\overline{f}_1$, then
  - find new X'_1
  - find $M_j^1\,,\ j = 1, \ldots, Q$ (possibly, approximately)
- $\ldots$

Remark: Redefining Ms may not necessary be done every single time an upper bound is updated, but for instance every kth time.