Stochastic Algorithms for Solving a Multiperiod Quantile-Based Portfolio Optimization Problem

Anthony Coache    François Watier

Department of Mathematics
Université du Québec à Montréal

Annual Meeting of the SSC ★ May 27, 2019 ★ Calgary
Coherent risk measures

Multiperiod portfolio optimization problem

Stochastic search algorithms

Performance of search methods
Coherent risk measures

- Coherent risk measures
- Multiperiod portfolio optimization problem
- Stochastic search algorithms
- Performance of search methods
Coherent risk measure

Consider two random variables $X$ and $Y$. A risk measure $\rho$ is said to be coherent (Delbaen, 2002) if

- (Addition of capital) For all $a \in \mathbb{R}$, $\rho(X + a) = \rho(X) - a$;

- (Diversification principle) $\rho(X + Y) \leq \rho(X) + \rho(Y)$;

- (Proportional risk) For all $t \geq 0$, $\rho(tX) = t\rho(X)$;

- (Sure gain) If $X \geq 0$, then $\rho(X) \leq 0$.

Variance and value-at-risk (Artzner et al., 1999) are not coherent risk measures.
Quantile-based risk measures

Many coherent risk measures are proposed in the literature, such as the conditional value-at-risk (Rockafellar and Uryasev, 2002) and expected-shortfall (Acerbi and Tasche, 2002).

We focus on the coherent risk measure based on the $\alpha$-quantile $q_\alpha$

$$\rho_\alpha(X) = \frac{-1}{\alpha} \left( \mathbb{E} \left[ X 1(X \leq q_\alpha(X)) \right] + q_\alpha(X) \left( \alpha - \mathbb{P} \left[ X \leq q_\alpha(X) \right] \right) \right)$$

$$= \frac{-1}{\alpha} \int_0^\alpha q_u(X) \, du.$$  

If the random variable $X$ is continuous, then $\rho_\alpha$ is equivalent to the expected-shortfall.
Multiperiod portfolio optimization problem

- Coherent risk measures
- Multiperiod portfolio optimization problem
- Stochastic search algorithms
- Performance of search methods
Illustration of the problem

Invest our initial wealth and control the expected terminal wealth;
Illustration of the problem

Invest our initial wealth and control the expected terminal wealth;
Minimize a **coherent risk measure**;
Invest our initial wealth and control the expected terminal wealth;
Minimize a coherent risk measure;
Adjust our strategy according to market fluctuations.
Why consider a binomial model?

- Discrete-time models are easier to handle.
- We can condition on specific wealth values since random variables only take a finite number of values.
- Binomial models can be generalized (e.g. trinomial model, multinomial model, etc.).

Under some assumptions, the binomial model provides a discrete-time approximation of the Black-Scholes model (Kim et al., 2016).
Parameters of the problem

We suppose that the interest rate of the riskless asset $r \equiv 0$ w.l.o.g., which means all rates of return are discounted.

- Initial wealth: $X_0$
- Wealth at the end of period $i$: $X_i$
- Expected terminal wealth: $X^*$
- Wealth amount invested in the risky asset at the beginning of period $i$: $\pi_i$
- Rate of return of the risky asset at the end of the period $i$: $R_i$
  - Probability of a high-reward rate of return: $p$
  - High-reward and low-reward rates of returns: $U$ and $L$
- Threshold for the risk measure: $\alpha$
- Number of periods: $n$
Multiperiod Portfolio Optimization Problem

Portfolio optimization problem with a binomial model

Definition of the problem

\[
\min_{\pi} \rho_\alpha(X_n) \quad \text{s.t.} \quad \mathbb{E}[X_n] = X^*.
\]

- **Self-financing constraint:**

\[
X_n = X_{n-1}(1 + r) + \pi_n(R_n - r) = X_0 + \sum_{i=1}^{n} \pi_i R_i.
\]

- One risky asset with independent rates of return over each period:

\[
R_i = \begin{cases} 
U & \text{with probability } p \\
L & \text{with probability } 1 - p 
\end{cases}, \quad \forall i = 1, \ldots, n.
\]

- \(U, L\) and \(p\) are chosen such that \(U > 0, L < 0\) and \(\mathbb{E}[R_i] > 0\).
Portfolio optimization problem with a binomial model

$\pi_1$ is a constant and $\pi_i, \ i = 2, \ldots, n$ are random variables. Thus there are $2^n$ possible terminal wealth values:

- $R_1 = U$ leads to $X_1^{(1)}$
- $R_1 = L$ leads to $X_1^{(2)}$
- $R_2 = U$ leads to $X_2^{(1)}$ and $X_2^{(3)}$
- $R_2 = L$ leads to $X_2^{(2)}$ and $X_2^{(4)}$
Where to look for the global minimum?

Proposition

The following risk measure is a convex function:

\[
\rho_\alpha(X) = \frac{-1}{\alpha} \left( E[X \mathbb{1}(X \leq q_\alpha(X))] + q_\alpha(X) (\alpha - P[X \leq q_\alpha(X)]) \right)
= \frac{-1}{\alpha} \int_0^\alpha q_u(X) du.
\]

The solution of this convex optimization problem is on the boundaries. The global minimum is necessarily obtain when the random variable \(X_n\) takes two unique values.
Where to look for the global minimum?

- Partition possible terminal wealth values into two groups;
- Solve both linear systems such that every wealth values in a group is equal;
- Compute $\rho_\alpha$ with these terminal wealth values.

Number of different combinations

$$\sum_{k=1}^{2^n-1-1} \binom{2^n}{k} + \frac{1}{2} \binom{2^n}{2^{n-1}} = 2^{2^n-1} - 1.$$ 

Stochastic search algorithms are essential to avoid enumerating each and every possible partitioning.

$(2^{2^6-1} - 1 \approx 9.2 \times 10^{18}$ partitions...)$
Stochastic search algorithms

- Coherent risk measures
- Multiperiod portfolio optimization problem
- Stochastic search algorithms
- Performance of search methods
Discrete uniform search algorithm (without replacement)

**Input:** Initialization of parameters ;

1. Initialize the number of iterations ;
2. Generate a discrete uniform without replacement sequence of integers;
3. Set the minimum risk measure to infinity ;
4. for $ii = 1$ to the number of iterations do
   5. Select the $ii$-th partitioning ;
   6. Compute its associated risk measure ;
   7. if it improves the minimum risk measure then
   8. Update the minimum risk measure found ;

**Output:** The minimum risk measure and its associated partitioning ;
Discrete uniform search algorithm (without replacement)

- This algorithm is efficient on average to find the global minimum, indeed
  \[ E[\text{Nbr. of iterations to find global minimum}] = 2^{2n-2}. \]
- It cannot visit the same combination twice.
- A big amount of memory space is required when the number of periods \( n \) grows. Its realization with \( n \geq 6 \) periods is almost impracticable on a single computer.

We then propose a stochastic algorithm that takes advantage of the structure of the problem and the binomial model.
Markovian change-when-improve algorithm

**Input:** Initialization of the number of iterations and parameters;
1. Select randomly a partitioning;
2. Compute its associated risk measure;
3. Create a memory variable;
4. for $ii = 2$ to the number of iterations do
   5. Select a label and remove it from the memory variable;
   6. Change partitioning by switching group this label;
   7. Compute the new associated risk measure;
   8. if it improves the minimum risk measure then
      9. Update the minimum risk measure found;
   10. if it improves the risk measure compared to last iteration then
       11. Update partitioning and memory variable;
   12. else if the memory variable is empty then
       13. Reinitialize partitioning and the memory variable;
       14. Compute its associated risk measure;
**Output:** The minimum risk measure and its associated partitioning;
Markovian change-when-improve algorithm

- It takes advantage of the structure of the problem.
- The Markovian change-when-improve search keeps track of at most the last $2^n$ combinations seen, which is less problematic when $n$ grows.
- This algorithm changes partitioning only when there is an improvement of the cost function.

Next slides illustrate the different steps and variables states of the Markovian change-when-improve algorithm for a portfolio optimization problem with $n = 3$ periods.
Markovian change-when-improve - Illustration

P1 (1st partition) [1 3]
P2 (2nd partition) [2 4 5 6 7 8]
P1Temp
P2Temp
FctTemp ($\rho_\alpha$) -0.2471
Memory [1 2 3 4 5 6 7 8]

FctLast ($\rho_\alpha$)
-0.2471
FctMin ($\rho_\alpha$)
-0.2471
CombinMin [1 3]
Global minimum -0.8595

Initialize combinations
Markovian change-when-improve - Illustration

P1 (1st partition)  [1 3]
P2 (2nd partition)  [2 4 5 6 7 8]
P1Temp  [1]
P2Temp  [2 3 4 5 6 7 8]
FctTemp ($\rho_\alpha$)  -0.5281
FctLast ($\rho_\alpha$)  -0.2471
FctMin ($\rho_\alpha$)  -0.2471
CombinMin  [1 3]
Memory  [1 2 3 4 5 6 7 8]

Split, remove from memory and compute the risk measure

Global minimum  -0.8595
Markovian change-when-improve - Illustration

P1 (1st partition) [1]
P2 (2nd partition) [2 3 4 5 6 7 8]
P1Temp [1]
P2Temp [2 3 4 5 6 7 8]
FctTemp ($\rho_\alpha$) -0.5281
Memory [1 2 3 4 5 6 7 8]

FctLast ($\rho_\alpha$)
-0.5281

FctMin ($\rho_\alpha$)
-0.5281

CombinMin [1]

Global minimum -0.8595

Update partitions and the global minimum
## Markovian change-when-improve - Illustration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FctLast ($\rho_\alpha$)</td>
<td>-0.5281</td>
</tr>
<tr>
<td>FctMin ($\rho_\alpha$)</td>
<td>-0.5281</td>
</tr>
<tr>
<td>CombinMin</td>
<td>[1]</td>
</tr>
<tr>
<td>Global minimum</td>
<td>-0.8595</td>
</tr>
</tbody>
</table>

P1 (1st partition)  
P2 (2nd partition)  
P1Temp  
P2Temp  
FctTemp ($\rho_\alpha$)  
Memory  

Split, remove from memory and compute the risk measure
Markovian change-when-improve - Illustration

P1 (1st partition) [1]
P2 (2nd partition) [2 3 4 5 6 7 8]
P1Temp [1 6]
P2Temp [2 3 4 5 7 8]
FctTemp ($\rho_{\alpha}$) -0.2471
Memory [1 2 3 4 5 6 7 8]

FctLast ($\rho_{\alpha}$)
-0.5281
FctMin ($\rho_{\alpha}$)
-0.5281
CombinMin [1]
Global minimum -0.8595

Keep partitions since we do not improve the risk measure
Markovian change-when-improve - Illustration

P1 (1st partition) [1]
P2 (2nd partition) [2 3 4 5 6 7 8]
P1Temp [1 2]
P2Temp [3 4 5 6 7 8]
FctTemp (\(\rho_\alpha\)) -0.2471
Memory [1 2 3 4 5 6 7 8]

FctLast (\(\rho_\alpha\)) -0.5281
FctMin (\(\rho_\alpha\)) -0.5281
CombinMin [1]
Global minimum -0.8595
Markovian change-when-improve - Illustration

P1 (1st partition) [1]
P2 (2nd partition) [2 3 4 5 6 7 8]
P1Temp [1 7]
P2Temp [2 3 4 5 6 8]
FctTemp (\(\rho_\alpha\)) -0.2471
Memory [1 2 3 4 5 6 7 8]

<table>
<thead>
<tr>
<th>FctLast ((\rho_\alpha))</th>
<th>-0.5281</th>
</tr>
</thead>
<tbody>
<tr>
<td>FctMin ((\rho_\alpha))</td>
<td>-0.5281</td>
</tr>
<tr>
<td>CombinMin</td>
<td>[1]</td>
</tr>
<tr>
<td>Global minimum</td>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Stochastic Search Algorithms

Markovian change-when-improve - Illustration

P1 (1st partition) [1]
P2 (2nd partition) [2 3 4 5 6 7 8]
P1Temp [1 3]
P2Temp [2 4 5 6 7 8]
FctTemp (\(\rho_\alpha\)) -0.2471
Memory [1 2 3 4 5 6 7 8]

<table>
<thead>
<tr>
<th>FctLast ((\rho_\alpha))</th>
<th>-0.5281</th>
</tr>
</thead>
<tbody>
<tr>
<td>FctMin ((\rho_\alpha))</td>
<td>-0.5281</td>
</tr>
<tr>
<td>CombinMin</td>
<td>[1]</td>
</tr>
<tr>
<td>Global minimum</td>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Markovian change-when-improve - Illustration

P1 (1st partition) [1]

P2 (2nd partition) [2 3 4 5 6 7 8]

P1Temp [1 8]

P2Temp [2 3 4 5 6 7]

FctTemp ($\rho_\alpha$) -0.3600

Memory [1 2 3 4 5 6 7 8]

FctLast ($\rho_\alpha$) -0.5281

FctMin ($\rho_\alpha$) -0.5281

CombinMin [1]

Global minimum -0.8595
## Markovian change-when-improve - Illustration

<table>
<thead>
<tr>
<th>P1 (1st partition)</th>
<th>[1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2 (2nd partition)</td>
<td>[2 3 4 5 6 7 8]</td>
</tr>
<tr>
<td>P1Temp</td>
<td>[1 4]</td>
</tr>
<tr>
<td>P2Temp</td>
<td>[2 3 5 6 7 8]</td>
</tr>
<tr>
<td>FctTemp ($\rho_\alpha$)</td>
<td>-0.2471</td>
</tr>
<tr>
<td>Memory</td>
<td>[1 2 3 4 5 6 7 8]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FctLast ($\rho_\alpha$)</th>
<th>-0.5281</th>
</tr>
</thead>
<tbody>
<tr>
<td>FctMin ($\rho_\alpha$)</td>
<td>-0.5281</td>
</tr>
<tr>
<td>CombinMin</td>
<td>[1]</td>
</tr>
<tr>
<td>Global minimum</td>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Markovian change-when-improve - Illustration

P1 (1st partition) \[1\]

P2 (2nd partition) \[2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8\]

P1Temp \[1 \ 5\]

P2Temp \[2 \ 3 \ 4 \ 6 \ 7 \ 8\]

FctTemp \(\rho_\alpha\) \(-0.2471\)

Memory \[1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8\]

FctLast \(\rho_\alpha\) \(-0.5281\)

FctMin \(\rho_\alpha\) \(-0.5281\)

CombinMin \[1\]

Global minimum \(-0.8595\)
Markovian change-when-improve - Illustration

P1 (1st partition) [1 5 8]
P2 (2nd partition) [2 3 4 6 7]
P1Temp
P2Temp
FctTemp ($\rho_\alpha$)
Memory [1 2 3 4 5 6 7 8]

<table>
<thead>
<tr>
<th>FctLast ($\rho_\alpha$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0036</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FctMin ($\rho_\alpha$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.5281</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CombinMin</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Global minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Markovian change-when-improve - Illustration

P1 (1st partition) \[ [1 \ 5 \ 8] \]

P2 (2nd partition) \[ [2 \ 3 \ 4 \ 6 \ 7] \]

P1Temp \[ [1 \ 8] \]

P2Temp \[ [2 \ 3 \ 4 \ 5 \ 6 \ 7] \]

FctTemp \( \rho \alpha \) \[ -0.3600 \]

Memory \[ [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8] \]

FctLast \( \rho \alpha \) \[ -0.0036 \]

FctMin \( \rho \alpha \) \[ -0.5281 \]

CombinMin \[ [1] \]

Global minimum \[ -0.8595 \]
Markovian change-when-improve - Illustration

P1 (1st partition) [1 8]
P2 (2nd partition) [2 3 4 5 6 7]
P1Temp [1 8]
P2Temp [2 3 4 5 6 7]
FctTemp ($\rho_\alpha$) -0.3600
Memory [1 2 3 4 5 6 7 8]

<table>
<thead>
<tr>
<th></th>
<th>FctLast ($\rho_\alpha$)</th>
<th>FctMin ($\rho_\alpha$)</th>
<th>CombinMin</th>
<th>Global minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.3600</td>
<td>-0.5281</td>
<td>[1]</td>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Markovian change-when-improve - Illustration

P1 (1st partition)  [1 8]
P2 (2nd partition)  [2 3 4 5 6 7]
P1Temp          [8]
P2Temp          [1 2 3 4 5 6 7]
FctTemp ($\rho_\alpha$) -0.8595
Memory         [1 2 3 4 5 6 7 8]

<table>
<thead>
<tr>
<th></th>
<th>FctLast ($\rho_\alpha$)</th>
<th>FctMin ($\rho_\alpha$)</th>
<th>CombinMin</th>
<th>Global minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.3600</td>
<td>-0.5281</td>
<td>[1]</td>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Markovian change-when-improve - Illustration

<table>
<thead>
<tr>
<th></th>
<th>P1 (1st partition)</th>
<th>[8]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P2 (2nd partition)</td>
<td>[1 2 3 4 5 6 7]</td>
</tr>
<tr>
<td></td>
<td>P1Temp</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>P2Temp</td>
<td>[1 2 3 4 5 6 7]</td>
</tr>
<tr>
<td>FctTemp</td>
<td>((\rho_\alpha))</td>
<td>-0.8595</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td>[1 2 3 4 5 6 7 8]</td>
</tr>
</tbody>
</table>

Global minimum is found

<table>
<thead>
<tr>
<th></th>
<th>FctLast ((\rho_\alpha))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.8595</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>FctMin ((\rho_\alpha))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.8595</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CombinMin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[8]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Global minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.8595</td>
</tr>
</tbody>
</table>
Performance of search methods

- Coherent risk measures
- Multiperiod portfolio optimization problem
- Stochastic search algorithms
- Performance of search methods
Performance of search methods

Comparison of the uniform without replacement (Unif w/o) and Markovian change-when-improve (MCWI) algorithms

- Number of iterations to find the global minimum
- Minimum found after a fixed number of iterations

Here are the parameters of the problem with the same notation as specified earlier.

- Initial wealth: $X_0 = 1$
- Expected terminal wealth: $X^* = 6/5$
- Probability of a high-reward rate of return: $p = 0.75$
- High-reward rate of return: $U = 1$
- Low-reward rate of return: $L = -2$
- Threshold for the risk measure: $\alpha = 0.3$
- Number of periods: $n$
Performance of Search Methods

Number of iterations to find the global minimum

\[ 2^{2^{2}} - 1 = 7 \]

Uniform w/o replace search expectation :

4 iterations

MCWI search estimated expectation :

8.60 iterations

**Figure:** Distribution of the number of iterations to find the global minimum - MCWI 2 periods
Performance of Search Methods

Number of iterations to find the global minimum

\[2^{3} - 1 - 1 = 127\]

Uniform w/o replace search expectation:

64 iterations

MCWI search estimated expectation:

25.79 iterations

\[\text{Estimation with 10000 replications} \quad \text{Max} = 165 \text{ iterations} \]

\[\text{Mean} = 25.79\]

\[\text{Figure: Distribution of the number of iterations to find the global minimum - MCWI 3 periods}\]
Performance of Search Methods

Number of iterations to find the global minimum

\[ 2^{2^4 - 1} - 1 = 32\,767 \]

Uniform w/o replace search expectation:

16,384 iterations

MCWI search estimated expectation:

98.53 iterations

Figure: Distribution of the number of iterations to find the global minimum - MCWI 4 periods
Performance of Search Methods

Number of iterations to find the global minimum

\[ 2^{2^5 - 1} - 1 \approx 2 \times 10^9 \]

Uniform w/o replace search expectation:

\[ \approx 1 \times 10^9 \text{ iterations} \]

MCWI search estimated expectation:

\[ 294.53 \text{ iterations} \]

Figure: Distribution of the number of iterations to find the global minimum - MCWI 5 periods

Estimation with 5000 replications
Max = 1484 iterations

Mean = 294.53
Minimum found after a fixed number of iterations

Best of 4 iterations

Uniform w/o replace search replications that found the global minimum:

57.07%

MCWI search replications that found the global minimum:

48.22%

Figure: Distribution of the minimum values found by both algorithms - 2 periods
Performance of Search Methods

Uniform (w/o replace) vs. Markovian CWI

**Figure:** Proportion of 10000 replications that found the global minimum for both methods - 2 periods

![Graph showing comparison between Uniform (w/o replace) and Markovian CWI methods.](image)
Performance of Search Methods

Minimum found after a fixed number of iterations

Best of 65 iterations

Uniform w/o replace search replications that found the global minimum:

51.38%

MCWI search replications that found the global minimum:

95.23%

Figure: Distribution of the minimum values found by both algorithms - 3 periods
Performance of Search Methods

Uniform (w/o replace) vs. Markovian CWI

Figure: Proportion of 10000 replications that found the global minimum for both methods - 3 periods
Minimum found after a fixed number of iterations

Best of 150 iterations

Uniform w/o replace search replications that found the global minimum:

0.37%

MCWI search replications that found the global minimum:

82.64%

Figure: Distribution of the minimum values found by both algorithms - 4 periods
Uniform (w/o replace) vs. Markovian CWI

Figure: Proportion of 5000 replications that found the global minimum for both methods - 4 periods
Conclusion

- Stochastic algorithms provide **efficient procedures** to find optimal (or near-optimal) solutions to optimization problems.

- Using the **structure of the problem** improves significantly the efficiency of search algorithms in multiperiod portfolio optimization problems.

- It could provide some insights on the potential optimal strategy in the continuous case with models such as the **Black-Scholes model**.
Acknowledgments and references

I am grateful to NSERC and FRQNT for their financial support throughout my Master’s degree.


Thank you!

*Slides and this beamer template are available on my website:*

www.anthonycoache.ca