Portfolio Optimization under Cumulative Prospect Theory: An Alternating Direction Method of Multipliers

Yifan Yan ∗  Rujun Jiang †  Yun Shi ‡  Xiangyu Cui §

October 7, 2022

Abstract

In this paper, we consider portfolio optimization under cumulative prospect theory (CPT). Existing methods for CPT optimization are only available under particular assumptions that may not hold in practice. We propose the first numerical method for solving CPT portfolio optimization based on historical asset return under a general incomplete market setting. Our method is an alternating direction method of multiplier (ADMM), which introduces an auxiliary variable to represent the historical return of the associated portfolio. The main difficulty in our ADMM is that one of its two subproblems involves optimization with the CPT utility subject to a chain constraint. We develop two methods to solve this subproblem. The first one is based on the philosophy of dynamic programming, and the second one is a variant of the well-known pooling-adjacent-violators algorithm. We further demonstrate the theoretical convergence of the proposed ADMM method and the two methods for solving the difficulty subproblem. Our numerical experiments demonstrate the effectiveness of the proposed method. Based on the proposed method, our empirical study with real data further demonstrates how the CPT’s parameters influence the investor’s investment behavior.

1 Introduction

Portfolio optimization is one of the most important fundamental questions in financial analysis, whose goal is to assign optimal weights to different risky assets in order to meet certain requirements. The seminar portfolio selection frameworks in neoclassical finance theory are the mean-variance model (Markowitz, 1952) and the expected utility theory model (EUT) (Von Neumann and Morgenstern, 1944). In the mean-variance model, the investors clearly define the return and risk of the portfolio, as well as the trade-off between return and risk. Along this direction, Konno and Suzuki (1995) extends the mean-variance model into the mean-variance-skewness model, and Konno and Koshizuka (2005) proposes the mean-absolute deviation model. Besides these risk measures relying on the whole distribution, people also propose tail risk measures, which rely on the tail of the distribution, such as...

∗School of Data Science, Fudan University, Shanghai, China, yanyf21@m.fudan.edu.cn
†Corresponding author. School of Data Science, Fudan University, Shanghai, China, rjjiang@fudan.edu.cn
‡Academy of Statistics and Interdisciplinary Sciences, Faculty of Economics and Management, East China Normal University, Shanghai, China, yshi@fem.ecnu.edu.cn
§School of Statistics and Management, Shanghai University of Finance and Economics, Shanghai, China, cui.xiangyu@mail.shufe.edu.cn
as the Value at Risk (VaR) (Jorion, 1997), Conditional VaR (Rockafellar et al., 2000), and weighted VaR (Wei, 2018).

Different from the mean-risk framework, the EUT model evaluates the goal as the weighted utilities of all the possible terminal wealth. The EUT model requires that the investor is uniformly risk-averse (i.e., the utility function is strictly concave), and the investor is rational in evaluating probabilities objectively. However, the EUT model fails to explain many market anomalies (Allais, 1953, Kahneman and Tversky, 1979). Thus, behavioral finance has been developing since the 1980s, which posits that emotion and psychology will influence investors and create room for bounded rationality. Among those developed behavioral models, the cumulative prospect theory (CPT) (Tversky and Kahneman, 1992) is the most important one. The CPT portfolio optimization model has four key elements: (i) the investors evaluate the portfolio on gains and losses with respect to a reference point, (ii) the investors are loss-averse, which means that the degree of suffering due to loss is more than the degree of happiness due to the same magnitude of gain, (iii) the investors are risk-averse for gains and risk-seeking for losses, and (iv) the investors have an inverse S-shaped probability weighting function, which gives more weights for the extreme losses or gains with a small probability. The first three elements imply that the investors have an S-shaped utility function, which is non-convex and non-smooth at the reference point. This, together with the fourth element that introduces probability distortion for different scenarios, makes the CPT portfolio optimization model difficult to solve.

The CPT portfolio optimization model has been studied in both the continuous time setting and the discrete time setting. Jin and Zhou (2008); He and Zhou (2011b); van Bilsen and Laeven (2020); Rüschendorf and Vanduffel (2020); Bi et al. (2021) are the representative works in the continuous time setting. They have very similar solution schemes. In general, the market is assumed to be complete, which has a unique martingale measure. By employing the martingale method, the CPT portfolio selection problem is transferred into a static optimization problem with the decision variable being the optimal terminal wealth. Then, the static optimization problem is solved by directly applying some numerical methods or transferring the decision variable into the quantile function of the optimal terminal wealth. Different from the continuous time setting, the market in the discrete time setting is often incomplete and the martingale method is not applicable. Most works in the literature solve the CPT problem under some particular assumptions, for example, normal distribution assumption (Barberis and Huang, 2008), one risky asset market (Bernard and Ghossoub, 2010; He and Zhou, 2011a; Zou and Zagst, 2017), elliptical distribution assumption (Pirvu and Schulze, 2012) and binomial tree assumption (Barberis and Xiong, 2009). There are also some works considering the multi-period CPT model, such as De Giorgi and Legg (2012); Shi et al. (2015), which mainly focus on the dynamic relationship between different periods. Numerical methods for CPT optimization are limited. Most of the studies only use heuristics (Levy and Levy, 2004; Hens and Mayer, 2014) or just use generic nonlinear optimizers (Srivastava et al., 2022). Very recently, Luxenberg et al. (2022) propose three methods for solving CPT optimization using its structure. However, their methods only apply to an approximation of the original CPT optimization problem by enforcing monotonicity on weights (see Section 2.2 in Luxenberg et al., 2022 for the discussion). To the best of our knowledge, the CPT optimization problem with a very general incomplete market setting remains unsolved, either analytically or numerically.

In this paper, we consider CPT optimization for multiple risky assets, whose distribution is characterized by $N$ historical scenarios, and the admissible portfolio strategy is assumed to satisfy
general convex constraints. The main difficulties of CPT optimization lie in two aspects, the S-shaped utility function is non-concave and non-smooth at the reference point, and the distorted weights for the utilities depend on whether the portfolio return is below or above the reference point. To handle these two issues, we develop a new alternating direction method of multipliers (ADMM) based on efficient subproblem solvers.

The ADMM is a widely used optimization method in non-smooth non-convex optimization if there is a certain separable structure in the objective function (Glowinski and Marrocco, 1977; Boyd et al., 2011). The ADMM method or its variant have been studied comprehensively in portfolio optimization literature; see, e.g., mean-VaR portfolio optimization (Cui et al., 2018), generalized risk parity portfolio optimization (Costa and Kwon, 2020), short-term sparse portfolio optimization (Lai et al., 2018) and fund of fund optimization (Chen et al., 2022).

We first reformulate the CPT maximization problem by introducing an auxiliary variable that represents the historical return of the portfolio. Then we develop an ADMM to solve this reformulation, using its special structure. One subproblem of the ADMM, which is a convex quadratic program, is easy to solve. The main difficulty lies in solving another subproblem, which has a non-convex separable objective and a chain constraint. We develop a dynamic programming (DP) method to solve this subproblem. We demonstrate that the DP method can find the global optimal solution. Despite the solidness in theory, the DP method may be slow in practice if there are a large number of historical scenarios of the asset returns. For practical efficiency, we further propose a two-partition pooling-adjacent-violators (TPPAV) algorithm to solve this subproblem, which can be seen as an extension of the celebrated pooling-adjacent-violators (PAV) algorithm (Ayer et al., 1955; Brunk et al., 1972; Best and Chakravarti, 1990; Ahuja and Orlin, 2001). We further prove that the TPPAV algorithm converges to a local minimizer of the subproblem. However, sometimes the TPPAV may not be a descent method, which destroys the convergence of the ADMM. To remedy this, we develop a hybrid method that combines the TPPAV method and the DP method, which is efficient in practice and guarantees the convergence of the ADMM. Our numerical study shows that the proposed ADMM always performs significantly better than the general solver fmincon in MATLAB.

We also conduct an empirical study based on 48 industry indices of US market and investigate the impacts of CPT’s parameters on the optimal portfolio. We choose the parameter setting in the literature as a benchmark. We find that by changing the reference point from 0 to risk-free return, the optimal portfolios and their performances almost have no changes. By removing probability distortion, the optimal portfolio becomes more diversified and also attains better performance in terms of Sharpe ratio. This finding on probability distortion is consistent with the theoretical prediction in Barberis (2013). Both increasing the risk aversion parameter and removing loss aversion make the investor less risk averse, therefore taking a more risky and gambling portfolio.

The remaining of this paper is organized as follows. In Section 2, we specify the CPT utility optimization problem. Then in Section 3, we propose the ADMM framework and deduce its convergence analysis. In Section 4, we propose two methods, the DP and the TPPAV methods, to solve one subproblem in the ADMM. In Section 5, we conduct numerical experiments to show the effectiveness of the proposed method. In Section 6, we conduct an empirical analysis to check how different parameters in CPT utility will affect investors’ behaviors. We conclude our paper in Section 7.
2 CPT Utility Optimization

In the CPT optimization model, the investors are loss-averse, risk-averse for gains, and risk-seeking for losses, which implies an S-shaped utility function \( (Tversky and Kahneman, 1992) \). We write the utility function as follows

\[
U(y) = \begin{cases} 
-\mu(B - y)^\alpha, & y \leq B, \\
(y - B)^\alpha, & y > B,
\end{cases}
\]

where \( B \) is the reference point to define gains and losses, \( \mu \) is the loss aversion parameter, and \( \alpha \) is the degree of risk aversion. Furthermore, the investors evaluate the probability for gains and losses through two probability distortion functions instead of objective probability, \( \omega_+(p) = \frac{p^\gamma}{(p^\gamma + (1 - p)^\gamma)^{1/\gamma}} \), \( \omega_-(p) = \frac{p^\delta}{(p^\delta + (1 - p)^\delta)^{1/\delta}} \), where \( p \) is the objective probability of an event, \( \gamma \) is the probability distortion parameter for gains, \( \delta \) is the probability distortion parameter for losses. The expected utility of the CPT optimization model is given as

\[
\int_B^{+\infty} (y - B)^\alpha d[\omega_+(1 - F_Y(y))] - \mu \int_{-\infty}^B (B - y)^\alpha d[\omega_-(1 - F_Y(y))],
\]

where \( F_Y(y) \) is the probability distribution function (CDF) of the terminal wealth achieved by portfolio \( x \). According to \( Tversky and Kahneman (1992) \), \( \mu = 2.25 \), \( \alpha = 0.88 \), \( \gamma = 0.61 \) and \( \delta = 0.69 \).

Let \( R \in \mathbb{R}^{N \times d} \) be the historical return matrix, which contains \( N \) scenarios of \( d \) risky assets. Each scenario has the probability \( 1/N \). Let \( x \) be the portfolio on \( d \) asset. Thus, \( Rx \) is the \( N \)-dimensional vector, which represents the scenarios of the portfolio’s return after a one-period investment. Let \( X \), which is a subset of \( \mathbb{R}^d \), be the set of all admissible portfolio strategies. For example, the no-shorting constraint \( X = \{ x \mid x \geq 0, e^T x = 1 \} \), or the upper and lower bound constraint \( X = \{ x \mid u \geq x \geq \ell, e^T x = 1 \} \), where \( 0 \) and \( e \) are the vectors with all zero and one elements, respectively, \( e^T x = 1 \) denotes the budget spending, \( ^T \) denote the transpose operator, and \( \ell \) and \( u \) are the lower and upper bounds.

Under the setting of \( N \) scenarios, the CPT utility maximization model can be cast as the following optimization problem,

\[
\max x \sum_{i=1}^N c_i U((Rx)_{[i]})
\]

\[
s.t. \quad x \in X,
\]

where \((Rx)_{[i]}\) denotes the \( i \)-th smallest scenario of the portfolio’s return, \( c_i \) denotes the distorted probability of \( i \)-th smallest scenario, which is defined as

\[
c_i = \begin{cases} 
\omega_\left( \frac{i}{N} \right) - \omega_\left( \frac{i-1}{N} \right), & (Rx)_{[i]} \leq B, \\
\omega_+\left( \frac{N - i + 1}{N} \right) - \omega_+\left( \frac{N - i}{N} \right), & (Rx)_{[i]} > B.
\end{cases}
\]
Note that $c_i > 0$ for all $i = 1, \ldots, N$. The main difficulty of the problem lies in two aspects. First, the utility function is nonconcave for the losses and non-smooth at the reference point. Second, the distorted probability depends on whether the portfolio’s return is below or above the reference point.

3 The ADMM Algorithm

In this section, we propose an alternating direction method of multipliers (ADMM) to solve the CPT optimization problem (1). Note that the main difficulty of optimizing (1) lies in the utility function, which is non-smooth and non-convex. At the first glance, the non-smoothness comes from the order statistics of $R\mathbf{x}$, which is the historical scenarios of the associate portfolio, and the non-convexity comes from the S-shape utility. However, we point out another more subtle issue is that the probability distortion (2) introduces additional non-convexity and non-smoothness that are related to the values of the portfolio and the reference point $B$.

Problem (1) is difficult to solve, even though the restriction $\mathcal{X}$ is usually a polyhedron. To handle the difficulty in the utility function, we introduce an auxiliary variable $y \in \mathbb{R}^n$ to represent the historical return of the portfolio. That is, we reformulate problem (1) as

$$\max_{\mathbf{x}, y} \sum_{i=1}^{N} c_i U(y[i]),$$

s.t. \hspace{0.5cm} \begin{align*}
    y &= R\mathbf{x}, \\
    \mathbf{x} &\in \mathcal{X}.
\end{align*}

The above reformulation has a linear equality constraint and can be solved by the ADMM method once its subproblems are shown able to be solved efficiently.

Now let us describe our ADMM framework. For consistency with optimization literature, we consider the following minimization reformulation of problem (3)

$$\min_{\mathbf{x}, y} - \sum_{i=1}^{N} c_i U(y[i]),$$

s.t. \hspace{0.5cm} \begin{align*}
    y &= R\mathbf{x}, \\
    \mathbf{x} &\in \mathcal{X}.
\end{align*}

The augmented Lagrangian function of the above problem is

$$L_\sigma(\mathbf{x}, y; \lambda) = - \sum_{i=1}^{N} c_i U(y[i]) + \frac{\sigma}{2} \left\| y - R\mathbf{x} + \frac{\lambda}{\sigma} \right\|_F^2 + I_\mathcal{X}(\mathbf{x})$$

where $I_\mathcal{X}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in \mathcal{X}, \\ \infty, & \text{else} \end{cases}$ is the indicator function of set $\mathcal{X}$, $\lambda$ is the Lagrange multiplier and $\sigma > 0$ is the quadratic penalty parameter. Note that it is difficult to simultaneously optimize $\mathbf{x}$ and $y$. This motivates us to adopt the ADMM framework that is widely used to handle a separable
Algorithm 1 ADMM for CPT utility maximization problem (4)

Input: $R$, $\sigma$, $\epsilon_1$, $\epsilon_2$, $y^0$, $\lambda^0$, and $k = 0$

Output: the portfolio weight $x^*$

1: repeat
2: Solve (6) to obtain $x^{k+1}$
3: Solve (11) to obtain $y^{k+1}$ either exactly or approximately
4: Update $\lambda^{k+1} = \lambda^k + \sigma(y^{k+1} - Rx^{k+1})$
5: $k = k + 1$
6: until $\|y^k - Rx^k\| < \epsilon_1$ and $\|y^k - y^{k-1}\| < \epsilon_2$

We summarise our ADMM framework in Algorithm 1.

It is obvious that the $x$-subproblem (5a) is a quadratic program when $X$ is a polyhedron,

$$\min_{x} \frac{\sigma}{2} \|y^k - Rx + \frac{\lambda^k}{\sigma}\|_F^2,$$

s.t. $x \in X$,

which is easy to solve by various existing solvers.

The main difficulty is solving the $y$-subproblem (5b), which is a non-convex and non-smooth optimization problem. Fortunately, by digging into the structure of the $y$-subproblem, we derive two efficient algorithms, whose details are given in Section 4. At the $k$th iteration, the $y$-subproblem is to minimize the following function

$$\Phi(y) = - \sum_{i=1}^{N} c_i U(y_{[i]}) + \frac{\sigma}{2} \|y - Rx^{k+1} + \frac{\lambda^k}{\sigma}\|_F^2.$$

Let $\Omega(y)$ denote the first component of $\Phi(y)$, i.e.,

$$\Omega(y) = - \sum_{i=1}^{N} c_i U(y_{[i]}).$$

Before deducing our convergence analysis of the proposed ADMM algorithm, let us first introduce some results from non-smooth analysis. We now show that both $\Phi(y)$ and $\Omega(y)$ are locally Lipschitz so that the stationary point can be characterized by the Clarke generalized gradient (Clarke, 1990), as long as $y$ satisfies $y_i \neq B$, $i = 1, \ldots, N$. In fact, in each iteration of the proposed ADMM algorithm, this condition is always satisfied; see Lemma 3 below.

Lemma 1. The functions $\Phi(y)$ and $\Omega(y)$ are both locally Lipschitz continuous in some neighbourhood of any given $y$ satisfying $y_i \neq B$, $i = 1, \ldots, N$. 
Proof. As \( \sum_{j=1}^{m} \frac{2}{c_j} \| y - R x + \frac{1}{c_j} \|^2 \) is differentiable, we only need to show that \( \Omega(y) \) is locally Lipschitz. Let \( B(y, \epsilon) \) denote the open ball with radius \( \epsilon \) centered at \( y \). First note that \( U(\theta) \) is locally Lipschitz for \( \theta \neq B \) as \( U(\theta) \) is differentiable at \( \theta \neq B \).

If \( y_{[i]} = \bar{y}_{[i+1]} \) for \( i = 1, 2, \ldots, N \), let
\[
\epsilon = \min \{ |B - \bar{y}_{[1]}|, |B - \bar{y}_{[N]}| \}.
\]
Otherwise let
\[
\epsilon = \max \left\{ 0, \min \left\{ |B - \bar{y}_{[1]}|, |B - \bar{y}_{[N]}|, \min_{i=1,\ldots,N-1} \{|\bar{y}_{[i+1]} - \bar{y}_{[i]}|\} \right\} \right\}.
\]
Let \( y^1, y^2 \in B(y, \epsilon) \) be two arbitrary points. Thanks to the definition of \( \epsilon \), there exist two permutations \( \{j_1, j_2, \ldots, j_N\} \) and \( \{k_1, k_2, \ldots, k_N\} \) of \( \{1, 2, \ldots, N\} \) such that
\[
y_{j_i}^1 \leq y_{j_{i+1}}^1, \quad \bar{y}_{j_i} \leq \bar{y}_{j_{i+1}}
\]
and
\[
y_{k_i}^2 \leq y_{k_{i+1}}^2, \quad \bar{y}_{k_i} \leq \bar{y}_{k_{i+1}}
\]
for \( i = 1, 2, \ldots, N \). (The two permutations may be different if \( y_{[i]} = y_{[i+1]} \) for some \( i \).) Therefore we have
\[
|\Omega(y) - \Omega(y^1)| = \left| \sum_{i=1}^{N} c_i U(\bar{y}_{j_i}) - U(y_{j_i}^1) \right|
\leq \sum_{i=1}^{N} c_i |U(\bar{y}_{j_i}) - U(y_{j_i}^1)|
\leq \max_{1 \leq i \leq N} c_i L_{j_i} |y_{j_i} - y_{j_i}^1|
\leq L^1 \|y - y^1\|,
\]
where \( L_{j_i} \) is the Lipschitz constant for \( U \) in some neighbourhood of \( \bar{y}_{j_i} \) and \( L^1 = \max_{1 \leq i \leq N} c_i L_{j_i} \). Similarly, we also have
\[
|\Omega(y) - \Omega(y^2)| \leq L^2 \|y - y^2\|
\]
for some constant \( L^2 \) that depends on \( \bar{y} \). This implies that
\[
|\Omega(y^1) - \Omega(y^2)| \leq |\Omega(y^2) - \Omega(y)| + |\Omega(y^1) - \Omega(y)| \leq (L^1 + L^2) \|y^1 - y^2\|.
\]
This completes the proof. \( \square \)

Our proof is fundamental and simple. The proof can also be used to show the Locally Lipschitz property for the empirical VaR in Cui et al. (2018), which can be represented as \( \sum_{i=1}^{N} a_i y_{[i]} \) for some \( a_i \geq 0 \), admitting a simpler form than \( \Omega(y) \).

As we have shown that \( \Omega(y) \) is locally Lipschitz for the every \( y \) as long as \( y_i \neq B \) \( \forall i \), the Clarke generalized gradient of \( \Omega(y) \), denoted by \( \partial \Omega(y) \), exists if \( y_i \neq B \) \( \forall i \) (Clarke, 1990). Let \( \chi(x) = \Omega(R x) \). By the chain rule, we have
\[
\partial \chi(x) \subset R^T \partial \Omega(R x).
\]
Denote by $N_X(\bar{x})$ the normal cone of $X$ at $\bar{x} \in X$, i.e.,

$$N_X(\bar{x}) = \{ z \in \mathbb{R}^d \mid \langle z, x - \bar{x} \rangle \leq 0, \forall x \in X \}.$$ 

It is well know that $\partial I_X(x) = N_X(\bar{x})$; see, e.g., (Beck, 2017, Example 3.5). We then have

$$\partial (\chi(x) + I_X(x)) \subset \partial \chi(x) + \partial I_X(x) = \partial \chi(x) + N_X(x),$$

where the first inclusion is due to Corollary 1 of (Clarke, 1990, Theorem 2.9.8). Suppose $x^*$ is a local minimizer of (5b). Then from (Clarke, 1990, Proposition 2.3.2) we have

$$0 \in \partial (\chi(x^*) + I_X(x^*)).$$

The above facts imply the following necessary optimality condition

$$0 \in R^T \partial \Omega(Rx^*) + N_X(x^*). \quad (7)$$

Due to (Clarke, 1990, Theorem 2.5.1), we obtain that

$$\partial \Omega(y) = \text{conv} \left( \{ \lim_{i \to \infty} \nabla f_i(y^i) : y^i \to y, y^i \notin S, \Omega \text{ is differentiable at } y^i \} \right),$$

where $\text{conv}(A)$ denotes the convex hull of set $A$ and $S$ is any set of Lebesgue measure 0 in $\mathbb{R}^N$. In fact, using the structure of $\Omega(y)$, we can characterize the expression of $\partial \Omega(y)$ explicitly by considering the coordinates that are equivalent. Let $\{i_1, i_2, \ldots, i_s\}$ be a permutation of $\{1, \ldots, N\}$ and $y$ be such that

$$y_{i_1} = \cdots = y_{i_{s_1}} < y_{i_{s_1 + 1}} = \cdots = y_{i_{s_2}} < \cdots < y_{i_{s_k - 1 + 1}} = \cdots = y_{i_{s_k}}.$$ 

By noting that the only non-differentiable points are those where at least two coordinates have the same value, we then have

$$\partial \Omega(y) = \text{conv} \left( \{ v : v_{i_j} \in \{-c_{ij}U(y_{i_j}) : j = s^t + 1, \ldots, s^{t+1}, t = 1, \ldots, k\} \} \right), \quad (8)$$

where we set $s^0 = 0$ for convention. For example, if $y = (y_1, y_2, y_3)$ with $y_2 = y_3 < y_1$, then noting

$$y_{[1]} = y_2, \ y_{[2]} = y_3, \ y_{[3]} = y_1 \quad \text{or} \quad y_{[1]} = y_3, \ y_{[2]} = y_2, \ y_{[3]} = y_1,$$

we have

$$\partial \Omega(y) = \text{conv} \left( \{ (-c_3U(y_1), -c_1U(y_2), -c_2U(y_3))^T, (-c_3U(y_1), -c_2U(y_2), -c_1U(y_3))^T \} \right).$$

We then further have the following lemma that is useful in our convergence analysis.

**Lemma 2** (Outer semicontinuity). For any sequences $\{v^k\}$ and $\{y^k\}$ such that $v^k \to v^*, \ y^k \to y^*$ and $v^k \in \partial \Omega(y^k)$, we have $v^* \in \partial \Omega(y^*)$.

**Proof.** The proof follows directly from (8). \qed

As the main theoretical result in this section, we will show that the proposed ADMM algorithm converges to a point satisfying the above necessary optimality condition under mild conditions. To this end, let us first make an assumption on the Lagrange multiplier.
Assumption 1. The sequence \( \{\lambda^k\} \) is bounded and satisfies the following condition

\[ \sum_{k=1}^{\infty} \|\lambda^{k+1} - \lambda^k\|^2 < \infty. \]

We remark that Assumption 1 is widely used in the ADMM literature; see, e.g., Xu et al. (2012), Bai et al. (2021) and even a stronger version in Shen et al. (2014).

We also make the following assumption on the \( x \)- and \( y \)-subproblem solutions.

Assumption 2. The \( x \)-subproblem is globally solved. The \( y \)-subproblem is solved such that

\[ L_\sigma(x^{k+1}, y^k; \lambda^k) - L_\sigma(x^{k+1}, y^{k+1}; \lambda^k) \geq 0, \]  

and

\[ 0 \in \partial \Omega(y^{k+1}) + \sigma \left( y^{k+1} - Rx^{k+1} + \frac{\lambda^k}{\sigma} \right). \]

We remark the above assumption is quite mild. Indeed, the \( x \)-subproblem is a quadratic program, which can be solved globally by many methods such as interior point methods (Nocedal and Wright, 1999). Meanwhile, the \( y \)-subproblem is solved to a stationary point \( y^{k+1} \) that has an objective value not larger than the starting point \( y^k \). We will propose subproblem solvers that can return a solution satisfying both (9) and (10) in Section 4.1.

Now we are ready to present the main convergence result for the ADMM algorithm.

Theorem 1. Suppose that \( R^TR \) is positive definite, \( \mathcal{X} \) is a convex and closed set, and Assumptions 1 and 2 hold. Let \( \{(x^k, y^k)\} \) be a sequence generated by the ADMM algorithm. Then any accumulation point of the sequence \( \{(x^k, y^k)\} \), denoted by \( (x^*, y^*) \), satisfies the following condition

\[ 0 \in R^T \partial \Omega(y^*) + N_\mathcal{X}(x^*), \quad y^* = Rx^*. \]

Thus we obtain that \( x^* \) is a stationary point of (1) in the sense that (7) holds.

The proof is given in Appendix A. We remark that the assumption on \( R^TR \) is mild as \( R^TR \) is indeed the estimated covariance matrix of assets. It is natural that a covariance matrix is positive definite when \( N > d \), i.e., the number of historical scenarios is larger than the number of assets. If we relax this assumption, we can add a proximal term in the \( x \)-subproblem, which modifies our algorithm to a proximal ADMM algorithm like Bai et al. (2021), and achieve a similar convergence result. For simplicity, we do not give details for this case. The assumption that \( \mathcal{X} \) is a convex and closed set is satisfied in practice, e.g., in many portfolio optimization applications, the weight has a box constraint or a simplex constraint (Cui et al., 2018).

4 Solving the \( y \)-Subproblem in the ADMM

In this section, we propose two methods to solve the \( y \)-subproblem (5b). Before presenting our methods, we first introduce a reformulation for the \( y \)-subproblem. Let \( w^{k+1} = Rx^{k+1} - \lambda^k/\sigma \). For simplicity, we omit the superscript in \( w^{k+1} \) in the following of this section. Let \( \{l_1, l_2, \ldots, l_N\} \) be
a permutation of \( \{1, \ldots, N\} \) such that \( w_1 \leq w_2 \leq \cdots \leq w_N \). As \( c_i U(y_{[i]}) \) is independent on any permutation of \( y \), the \( y \)-subproblem (5b) is equivalent to

\[
\min_y \sum_{i=1}^{N} -c_i U(y_i) + \frac{\sigma}{2} (y_i - w_i)^2 \\
\text{s.t. } y_1 \leq y_2 \leq \cdots \leq y_N.
\] (11)

The constraint \( y_1 \leq y_2 \leq \cdots \leq y_N \) is known as the simple chain constraint (Best et al., 2000). Such a constraint first occurred in isotonic regression (Ayer et al., 1955) and is widely studied in the literature (Brunck et al., 1972; Strömbärg, 1991). We remark that a similar reformulation idea was previously used in (Cui et al., 2018, Lemma 3). Without loss of generality, in this section we assume \( w_1 \leq w_2 \leq \cdots \leq w_N \) for further notational simplicity. Thus problem (11) is equivalent to the isotonic program

\[
\min_y \sum_{i=1}^{N} -c_i U(y_i) + \frac{\sigma}{2} (y_i - w_i)^2 \\
\text{s.t. } y_1 \leq y_2 \leq \cdots \leq y_N.
\] (12)

Next let us study several useful features of problem (12). For ease of notation, let \( g_i(y) \) be the negative of the weighted utility function

\[
g_i(y_i) = \begin{cases} 
a_i \mu (B - y_i)^\alpha, & y_i \leq B, \\
-b_i (y_i - B)^\alpha, & y_i > B,
\end{cases}
\]

where \( a_i, b_i > 0 \) are calculated from the probability distortion function (2). Let

\[
f_i(y_i) = g_i(y_i) + \frac{\sigma}{2} (y_i - w_i)^2.
\] (13)

Then problem (12) can be rewritten as

\[
\min \sum_{i=1}^{N} f_i(y_i) \\
\text{s.t. } y_1 \leq y_2 \leq \cdots \leq y_N.
\] (14)

We will utilize the structure of \( f_i \) to design our DP algorithm.

To this end, we first consider the property of the following function

\[
\zeta(y) = \begin{cases} 
a \mu (B - y)^\alpha + \frac{\sigma}{2} (y - w)^2, & y < B, \\
-b(y - B)^\alpha + \frac{\sigma}{2} (y - w)^2, & y > B,
\end{cases}
\] (15)

which is exactly in the form of \( f_i \). Here \( a, b, \sigma > 0, \mu > 1, \) and \( 0 < \alpha < 1 \). The first order derivative of \( \zeta(y) \) is

\[
\zeta'(y) = \begin{cases} 
-\alpha a \mu (B - y)^{\alpha-1} + \sigma (y - w), & y < B, \\
-\alpha b(y - B)^{\alpha-1} + \sigma (y - w), & y > B,
\end{cases}
\] (16)
and the second order derivative of $\zeta(y)$ is
\[
\zeta''(y) = \begin{cases} 
- \alpha(1 - \alpha)a \mu(B - y)^{\alpha - 2} + \sigma, & y < B, \\
\alpha(1 - \alpha)b(y - B)^{\alpha - 2} + \sigma, & y > B.
\end{cases}
\]

We first summarise the property of $\zeta(y)$ in interval $[B, \infty)$.

**Proposition 1.** In interval $[B, \infty)$, the function $\zeta(y)$ is convex, and has a unique local minimizer, denoted by $\xi$. Moreover, $\zeta'(y) < 0$ for $y \in (B, \xi)$, and $\zeta'(y) > 0$ for $y \in (\xi, \infty)$.

**Proof.** The convexity follows from $\zeta''(y) > 0$ for all $y \in (B, \infty)$ and the right continuity of $\zeta(y)$ at $B$. As $0 < \alpha < 1$, it follows from (16) that
\[
\lim_{y \to B^+} \zeta'(y) = -\infty, \quad \text{and} \quad \lim_{y \to \infty} \zeta'(y) = \infty.
\]
Hence $\zeta'(y)$ is strictly increasing in $(B, \infty)$, and thus there exists a unique point $\xi$ such that $\zeta'(\xi) = 0$. The strictly increasing property of $\zeta'(y)$ further implies that $\zeta'(y) < 0$ for $y \in (B, \xi)$ and $\zeta'(y) > 0$ for $y \in (\xi, \infty)$. \qed

The case when $y < B$ is summarised in the following proposition.

**Proposition 2.** When $y < B$, there exists exactly one zero point of $\zeta''(y)$, denoted by $c = B - \left(\frac{\mu(1-\alpha)a}{\sigma}\right)\frac{1}{\alpha-2}$.
Moreover, $\zeta''(y) > 0$ and thus $\zeta(y)$ is concave, in $y \in (-\infty, c)$; $\zeta''(y) < 0$ and thus $\zeta(y)$ is convex, in $y \in (c, B)$. For $\zeta'(y)$ in $(-\infty, B)$, we have the properties:

- **Case I:** $\zeta'(c) > 0$, and thus $\zeta'(y)$ has two zero points. In this case, $\zeta(y)$ has one local minimizer, which is the left zero point, and one local maximizer.

- **Case II:** either i) $\zeta'(c) < 0$, and thus $\zeta'(y)$ has no zero point, or ii) $\zeta'(c) = 0$, which is the only zero point of $\zeta'(y)$.

The proofs for the above proposition are similar to that of Proposition 1 and thus omitted to save space. See Figures 1 for illustrations of the two cases mentioned in the above two propositions.

![Figure 1: Illustration of graphs of $\zeta(y)$](image)
In both cases, the points $c$ and $B$ separate the graph of the function into three parts: the function is convex in the first part, concave in the second part, and convex in the third part.
Note that finding a local minimizer (maximizer, respectively) of a univariate convex (concave, respectively) function in the interval \((\ell, u)\), which is assumed to exist, is equivalent to finding a zero point of its derivative. This is a root-finding problem for monotone and continuous functions, and can be easily solved by various methods like binary search or Fibonacci search to very high accuracy; see, e.g., (Bazaraa et al., 2006, Section 8.2). In our implementation, we adopt binary search that maintains a search interval including the optimal solution and perform one function evaluation to reduce the interval by half in each iteration. We also remark that assuming \(G(\theta)\) is a monotone and continuous function in interval \((\ell, u)\), it takes at most \(O(\log_2(\frac{u-\ell}{\epsilon}))\) function evaluations to find an approximate root \(\bar{\theta}\) such that \(|\bar{\theta} - \theta^*| < \epsilon\), where \(\theta^*\) is a root for \(G(\theta) = 0\) in \((\ell, u)\) (Ahuja and Orlin, 2001). In practice, several tens of binary searches are sufficient to find a solution up to machine accuracy. Therefore, in this paper we always assume that there is an oracle that finds the exact root of a monotone and continuous function.

**Definition 1** (Root-finding oracle). A root-finding oracle is a procedure that either finds the root or claims that there is no root for a univariate function that is monotone and continuous in a given interval.

Therefore, if \(F(\theta)\) is a convex (concave, respectively) continuously differentiable function in \((\ell, u)\), then the root-finding oracle for \(F'(\theta)\) can find the local minimizer (maximizer, respectively).

We next present a useful result that the optimal solution of \(y\)-subproblem must satisfy a condition so that the Clarke generalized gradient is well defined according to Lemma 1, which plays an essential role in our algorithm design and the associated convergence analysis.

**Lemma 3.** Let \(y^*\) be an optimal solution of problem (14). Then we have \(y^*_i \neq B, \forall i = 1, 2, \ldots, N\).

**Proof.** It is straightforward to verify that

\[
\lim_{y \to B^-} f'_i(y) = \lim_{y \to B^+} f'_i(y) = -\infty, \quad \forall i.
\]

Suppose for some index \(i\) we have \(y^*_i = B\). Let \(j\) be the smallest index such that \(j \geq i\) and \(y^*_j > B\). Let \(\bar{y}_k = \min(B + \epsilon, y^*_j)\) for \(i \leq k \leq j\), where \(\epsilon\) is chosen such that \(f'_k(y) < 0\) for all \(y \in (B, B + \epsilon)\), and \(\bar{y}_k = y^*_k\) for all other \(k\). Then \(\bar{y}\) yields a smaller objective value and still satisfies the chain constraint, which contradicts the optimality of \(y^*\).

The next lemma shows that the optimal solution of problem (14) is bounded by known constants. This is a useful property for our algorithm convergence analysis.

**Lemma 4.** Let \(y^*\) be an optimal solution of problem (14). Then we have

\[
w_1 \leq y_1^* \leq \cdots \leq y_N^* \leq \max_{i = 1, \ldots, N} \{\xi_i\},
\]

where \(\xi_i\) is the unique local minimizer of \(f_i(\cdot)\) in \((B, \infty)\), \(i = 1, \ldots, N\).

**Proof.** First note that \(f_i(\cdot)\) is in the form of \(\zeta_i(\cdot)\). Note also that \(y_1^* \leq \cdots \leq y_N^*\) due to the constraint in (14). Suppose that \(y_1^* \leq \cdots \leq y_k^* < w_1 \leq y_{k+1}^*\). Define \(\bar{y}_i = \max(w_1, y_i^*), \ i = 1, \ldots, N\), which is also feasible to (14). It follows from (16) that \(f'_i(u) < 0\) for all \(u \in (-\infty, w_1), i = 1, \ldots, k\). This yields \(\sum_{i=1}^{N} f_i(y_i^*) < \sum_{i=1}^{N} f_i(\bar{y}_i)\), which contradicts the optimality of \(y^*\). Hence we must have

\[
w_1 \leq y_1^* \leq \cdots \leq y_N^*.
\]

The upper bound could be analyzed similarly using the fact that \(f'_i(u) > 0\) for all \(u \in (\xi_i, \infty)\), thanks to Proposition 1. \(\square\)
In the following of this paper, we set 
\[ l_b = w_1 \text{ and } u_b = \max \left( B + 1, \max_{i=1,\ldots,N} \{w_i\} + \alpha \max_{i=1,\ldots,N} \{b_i\} \frac{1}{\sigma} \right). \]
The above \( l_b \) and \( u_b \) serve as initial lower and upper bounds for the solutions for the following proposed algorithms for the \( y \)-subproblem. We claim that \( \xi_i < \max(B + 1, w_i + \frac{\alpha b_i}{\sigma}), \ i = 1, \ldots, N, \) and hence \( u_b \) is an upper bound for \( y_N^* \) due to Lemma 4. Indeed, letting \( u = \max(B + 1, w_i + \frac{\alpha b_i}{\sigma}) \), we have from (13) that 
\[ f_i'(u) = -\alpha b_i (u - B)^{\alpha - 1} + \sigma (u - w_i) \geq -\frac{\alpha b_i}{u - B} + \sigma (u - w_i) \geq -\alpha b_i + \sigma \frac{\alpha b_i}{\sigma} = 0. \]
Since \( u > B \) and \( \xi_i \) is the unique local minimizer of \( f_i(\cdot) \) in \([B, \infty)\), it follows that \( \xi_i < u \). Then we have
\[ \max_{i=1,\ldots,N} \{\xi_i\} < \min \left( B + 1, \max_{i=1,\ldots,N} \{w_i\} + \alpha \max_{i=1,\ldots,N} \{b_i\} \frac{1}{\sigma} \right). \]

### 4.1 Dynamic Programming Algorithm

In this section, we use the philosophy of dynamic programming (DP) to derive an algorithm that globally solves problem (14). Our method is motivated by the DP method for generalized nearly isotonic optimization proposed in Yu et al. (2020). Note that the method in Yu et al. (2020) is only applicable for convex separable objective functions and thus cannot apply to our case. The main advantage of our DP method is that it finds the exact solution for our non-convex objective function, by fully exploiting the problem structure.

Define \( h_1(z) = 0 \), and define \( h_n(z) \) recursively as 
\[ h_n(z) = \min_{y \leq z} f_{n-1}(y) + h_{n-1}(y), \quad n = 2, \ldots, N + 1. \] (17)
By definition, \( h_n(z) \) is a non-increasing function of \( z \) as a bigger \( z \) enlarges the feasible region of the minimization problem in (17). Let \( y^* \) be the optimal solution of (14). Noting (17) is equivalent to,
\[ h_{N+1}(y_{N+1}) = \min_{y_{N} \leq y_{N+1}} \left\{ f_N(y_N) + \min_{y_{N-1} \leq y_N} \left\{ f_{N-1}(y_{N-1}) + \cdots + \min_{y_1 \leq y_2} \{ f_1(y_1) + h_1(y_1) \} \right\} \right\}, \]
we can verify that (14) is equivalent to
\[ \min_{u \leq y_{N+1}} f_N(u) + h_N(u). \] (18)
for any given \( y_{N+1} \geq y_N^* \). Problem (18) can be seen as a univariate unconstraint optimization problem. The recursion (17) also gives a way to obtain an optimal solution once some upper bound of \( y_N^* \) is known (e.g., \( u_b \)), and the explicit forms of \( h_n \) by
\[ y_{n-1}^* = \arg\min_{y \leq y_n^*} f_{n-1}(y) + h_{n-1}(y), \quad n = N + 1, N, \ldots, 2. \] (19)
where \( y_{N+1}^* \) can be any upper bound of \( y_N^* \). Then \( y^* = (y_1^*, \ldots, y_N^*)^T \) is an optimal solution to the \( y \)-subproblem (14).

An outline of our DP algorithm is as follows: We first forward derive the explicit expressions for \( h_2, h_3, \ldots, h_{N+1} \). Then we backward using (19) to recover the optimal \( y_N^*, y_{N-1}^*, \ldots, y_1^* \). The details are derived in the next subsection.
4.1.1 Details of the Dynamic Programming Algorithm

A key observation of our DP algorithm is that $h_n$ is a piecewise function where in each piece $h_n$ is either a constant or a strictly decreasing function in a special form. Based on this observation, the DP algorithm uses a forward update to derive the expression of $h_n$, based on the expression of $h_{n-1}$, by decomposing the interval $[l_b, B]$ (or $[B, u_b]$) into consecutive intervals so that the expression of $h_n$ is unified in each interval for $n = 1, 2, \ldots, N + 1$.

Let us first consider the derivation of $h_n$ in $[l_b, B]$. We will show that $h_{n-1}(z)$ is a piecewise function with $K_n^L$ consecutive intervals, say $[s_k^{n-1}, s_k^{n-1}]$, $k = 1, 2, 3, \ldots, K_n^L$, which form the interval $[l_b, B]$, and also that the explicit expression of $h_{n-1}$ in each piece is in some unified form. Our proof is by induction.

To find the expression of $h_n(z)$ in $[s_k^{n-1}, s_k^{n-1}]$, we need first explore the explicit formula for an auxiliary function defined by

$$\phi_{n-1}(z) = f_{n-1}(z) + h_{n-1}(z),$$

for every interval $[s_k^{n-1}, s_k^{n-1}]$, $k = 1, 2, 3, \ldots, K_n^L$. This yields

$$h_n(z) = \min_{x \leq z} \phi_{n-1}(x), \quad z \in [s_k^{n-1}, s_k^{n-1}],$$

for $k = 1, \ldots, K_n^L$.

Specifically, suppose we have the expression of $h_{n-1}(z)$ in intervals $[s_1^{n-1}, s_2^{n-1}], \ldots, [s_{K_n^L}^{n-1}, s_2^{n-1}]$. We can derive the expression of $h_n(z)$ in $[s_k^{n-1}, s_k^{n-1}]$ by

$$h_n(z) = \min \left( \min_{x \leq s_k^{n-1}} \phi_{n-1}(x), \min_{s_k^{n-1} \leq x \leq z} \phi_{n-1}(x) \right), \quad z \in [s_k^{n-1}, s_k^{n-1}],$$

$$= \min \left( h_n(s_k^{n-1}), \min_{s_k^{n-1} \leq x \leq z} \phi_{n-1}(x) \right), \quad z \in [s_k^{n-1}, s_k^{n-1}].$$

where the second equation follows directly from the definition of $h_{n-1}$ in (17).

Repeating the update (22) for each interval $[s_k^{n-1}, s_k^{n-1}]$, we obtain the full expression for $h_n$ in the interval $[l_b, B]$. From (21) we will show that $h_n(z)$ is piecewise function, which is either strictly decreasing or constant in each piece.

To facilitate our analysis, we give a lemma on the continuity of $h_n(z)$, which follows from (17), the continuity of $f_i$, and $h_1(z) = 0$.

**Lemma 5.** The function $h_n(z)$ is a continuous function in $(-\infty, \infty)$.

As $f_i(\cdot)$ has different expressions in $(-\infty, B]$ and $[B, \infty)$, it is natural that the expression of $h_n(z)$ is split by $B$. Due to Lemma 4, the solutions must be in the interval $[l_b, u_b]$. For the sake of conciseness, we only show details on update of $h_n(z)$ in $[l_b, B]$, because the update is similar in $[B, u_b]$. We denote by $L_n$ and $R_n$ the set consisting of the consecutive pieces of $h_n$ in $[l_b, B]$ and $[B, u_b]$, respectively, such that $h_n$ admits a unified expression in each piece. Let $K_n^L$ and $K_n^R$ denote the numbers of intervals for $L_n$ and $R_n$, respectively.

To better facilitate the deduction of $h_n(z)$, we introduce three key procedures in our methods. The first procedure is used to decompose each piece of $\phi_{n-1} = f_{n-1} + h_{n-1}$ to smaller consecutive intervals with desirable properties.
**Definition 2 (DECOMPOSE).** For a function in the form of \( \zeta(\cdot) \) in an interval \([p, q] \subset [b, B]\). If \( \zeta(\cdot) \) is neither strictly decreasing nor strictly increasing in \([p, q]\), then we decompose \([p, q]\) into several smaller intervals where in each new interval \( \zeta(\cdot) \) is either strictly increasing or strictly decreasing. We refer to this procedure as DECOMPOSE. It outputs a set of consecutive intervals, i.e., \( T = \text{DECOMPOSE}([p, q], \zeta(\cdot)) \).

The well-definedness of DECOMPOSE follows from the structure of \( \zeta(\cdot) \). Now let us illustrate how DECOMPOSE works. Suppose \( \zeta(y) \) is neither strictly decreasing nor strictly increasing in \([p, q]\). Indeed, from Proposition 2, we see that there are three cases for \([p, q]\) depends on if the interval \((p, q)\) contains the local minimizer, denoted by \( m_1 \), and/or local maximum, denoted by \( m_2 \), and all the cases corresponds to Case I (see Figure 1).

1. \( \zeta(y) \) first strictly decrease, and then strictly increase. This scenario occurs if \((p, q)\) contains the local minimizer. We decompose \([p, q]\) as \( T = \{[p, m_1], [m_1, q]\} \) so that \( \zeta(y) \) is strictly decreasing in the first interval, and is strictly increasing in the second interval.

2. \( \zeta(y) \) first strictly decreases, then strictly increases, and finally strictly decreases. This scenario occurs if \((p, q)\) contains both the local minimizer and maximizer, in which case we must have \( p < c < q \) for \( c \) satisfying \( \zeta''(c) = 0 \) as discussed in Proposition 2. We decompose \([p, q]\) as \( T = \{[p, m_1], [m_1, m_2], [m_2, q]\} \), and \( \zeta(y) \) is strictly decreasing, strictly increasing and strictly decreasing in the three intervals, respectively.

3. \( \zeta(y) \) first strictly increases, and then strictly decreases. This scenario occurs if \((p, q)\) contains the local maximizer. We decompose \([p, q]\) as \( T = \{[p, m_2], [m_2, q]\} \) so that \( \zeta(y) \) is strictly increasing in the first interval, and is strictly decreasing in the second interval.

Note that the local minimizer or maximizer corresponds to the root of \( \zeta'(\cdot) \) in the associated interval, and thus can be found using our root-finding oracle.

**Remark 1.** If \([p, q]\) is an interval in \([B, ub]\), and \( \zeta(y) \) is neither strictly increasing nor strictly decreasing, then \((p, q)\) must contain the unique local minimizer, denoted by \( m_3 \), and we decompose \([p, q]\) as \( T = \{[p, m_3], [m_3, q]\} \).

The next procedure updates the expression of \( h_n \) using the philosophy of (22).

**Definition 3 (UPDATE).** Let \( \gamma(\cdot) \) be a strictly monotone and continuous function defined in the interval \([p, q]\) and \( \kappa \) be a constant such that \( \kappa \leq \gamma(p) \). Define \( \theta(z) = \min\{\kappa, \min_{y \leq z} \gamma(y)\} \). The procedure UPDATE aims to get an explicit piecewise expression of \( \theta(z) \) in the interval \([p, q]\). It works as \( \{S, \theta(\cdot)\} = \text{UPDATE}([p, q], \gamma(\cdot), \kappa) \), where \( S \) denotes the consecutive intervals that support \( \theta(\cdot) \).

When \( \gamma(y) \) is strictly increasing, because of \( \min_{y \leq z} \gamma(y) = \gamma(p) \geq \kappa \), from the definition of \( \theta(z) \), we have \( \theta(z) = \kappa, z \in [p, q] \), and \( S = \{[p, q]\} \). The case that \( \gamma(y) \) is a constant is trivial. We next show all possible outputs of this procedure in the nontrivial case that \( \gamma(y) \) is strictly decreasing. There are three scenarios.

1. If \( \gamma(p) = \kappa \), then \( \theta(z) = \gamma(z) \), \( z \in [p, q] \) and \( S = \{[p, q]\} \).

2. If \( \gamma(q) \geq \kappa \), then \( \theta(z) = \kappa, z \in [p, q], \) and \( S = \{[p, q]\} \) since \( \gamma(\cdot) \) is strictly decreasing.
3. If $\gamma(p) > \kappa > \gamma(q)$, then $\theta(z)$ is a piecewise function with two pieces. Indeed, the strictly decreasing property and continuity of $\gamma(\cdot)$ guarantees that exits a unique point in $(p,q)$, denoted by $m_4$, such that $\gamma(m_4) = \kappa$. Thus we have $\theta(z) = \begin{cases} \kappa, & z \in [p,m_4], \\ \gamma(z), & z \in [m_4,q], \end{cases}$ and $S = \{[p,m_4],[m_4,q]\}$.

In our algorithm, UPDATE is used to update the expression of $h_n$ (with $\gamma = \phi_{n-1}$) in each piece. We also remark that the time cost of UPDATE is dominated by computing a root for a monotone equation, which can be done by our root-finding oracle.

Now let us show that $h_n$ admits a unified expression in each piece using the two procedures DECOMPOSE and UPDATE.

**Lemma 6.** For each $m = 1, \ldots, N$ and $z \in [b,B]$, $h_m(z)$ can be characterized by a piecewise function with $K_m$ pieces, given by

$$h_m(z) = D_{m,k} \left( \mu a_{m,k}(B - z)^{\alpha} + \frac{\sigma}{2}(z - w_{m,k})^2 \right) + M_{m,k}, \quad z \in [s_k^{m},s_{k+1}^{m}], \quad k = 1, 2, \ldots, K_m$$

where $s_k^0 = b$, $s_{K_m}^m = B$, $D_{m,k}$ is a nonnegative integer, and all other constants are determined by problem (14).

**Proof.** We prove the lemma using induction. First note that $h_1$ can be expressed in (23) as $h_1(z) = 0$ by definition. Next we assume for $m = n - 1$, (23) holds.

Now let us show (23) holds for $m = n$. It is straightforward from (20) and the fact that $f_i$ is of expression (15) that $\phi_{n-1}(z)$ has the same expression as (23),

$$\phi_{n-1}(z) = \hat{D}_{n-1,k} \left( \mu \hat{a}_{n-1,k}(B - z)^{\alpha} + \frac{\sigma}{2}(z - \hat{w}_{n-1,k})^2 \right) + \hat{M}_{n-1,k}, \quad z \in [s_k^{n-1},s_{k+1}^{n-1}],$$

where $\hat{D}_{n-1,k} = D_{n-1,k} + 1$, $\hat{a}_{n-1,k} = \frac{D_{n-1,k} a_{n-1,k} + a_{n-1}}{D_{n-1,k}}$, $\hat{w}_{n-1,k} = \frac{D_{n-1,k} w_{n-1,k} + w_{n-1}}{D_{n-1,k}}$, and $\hat{M}_{n-1,k} = M_{n-1,k} + \frac{\sigma}{2} \left( D_{n-1,k} w_{n-1,k}^2 + w_{n-1,k}^2 - D_{n-1,k} \hat{w}_{n-1,k}^2 \right)$.

Hence $\phi_{n-1}(z)$ is in the form of $\zeta(y)$, defined in (15), plus a constant. Let

$$T_{k}^{n-1,L} = \text{DECOMPOSE}([s_k^{n-1},s_{k+1}^{n-1}],\phi_{n-1}(\cdot)).$$

For every smaller interval $[t_i,t_{i+1}]$ in $T_k^{n-1,L}$, $\phi_{n-1}$ is either strictly increasing or strictly decreasing due to Definition 2.

Then we deduce $h_n(z)$ sequentially for every $[t_i,t_{i+1}]$ in $T_k^{n-1,L}$ using the UPDATE procedure. Note that $h_n(t_i)$ is known when we proceed in $[t_i,t_{i+1}]$ as a result of the previous interval. When $t_i = s_k^{n-1} = b$, which is the very beginning of all intervals, we assign $h_n(t_i) = \phi_{n-1}(b)$, as $\phi_{n-1}$ is strictly decreasing in $(-\infty, b)$ due to Lemma 4 and $\phi_{n-1}$ is indeed the addition of several strictly decreasing function in $(-\infty, b)$. It follows from (21) that $h_n(t_i) \leq \phi_{n-1}(t_i)$. Then using UPDATE($[t_i,t_{i+1}],\phi_{n-1}(\cdot),h_n(t_i)$) we obtain the expression of $h_n$ in $[t_i,t_{i+1}]$. Specifically, from the UPDATE step we know that $h_n(\cdot)$ is a piecewise function that adopts the form of either $\phi_{n-1}(\cdot)$ or a constant in each piece. Thus $h_n(z)$ takes the form of (23) for $z \in [t_i,t_{i+1}]$. By sequentially update $h_n(z)$ in each interval of $[s_k^{n-1},s_{k+1}^{n-1}]$, we obtain the explicit formula of $h_n(z)$ for $z \in [b,B]$. □
Algorithm 2: The deduction of $h_n(\cdot)$ from $h_{n-1}(\cdot)$ in $[l_b, B]$

**Input:** $h_{n-1}(\cdot)$, $f_{n-1}(\cdot)$, and $L^{n-1} = \{[s_1^{n-1}, s_2^{n-1}], [s_2^{n-1}, s_3^{n-1}], \ldots, [s_{K_{n-1}^R}^{n-1}, s_{K_{n-1}^{K_{n-1}^R}+1}^{n-1}]\}$ with $s_1^{n-1} = l_b$

and $s_{K_{n-1}^R+1}^{n-1} = B$

**Output:** $\{h_n(\cdot), L^n\}$

1: $h_n(s_1^{n-1}) = \phi_{n-1}(s_1^{n-1})$ \text{ (20)}
2: $L^n = \emptyset$
3: for $k = 1, 2, \ldots, K_{n-1}^L$ do
4: \hspace{1em} $T_k^{n-1,L} = \text{DECOMPOSE}([s_k^{n-1}, s_{k+1}^{n-1}], \phi_{n-1}(\cdot))$
5: \hspace{1em} for $i = 1, \ldots, \text{Card}(T_k^{n-1,L})$ do
6: \hspace{2em} select $[t_i, t_{i+1}] \in T_k^{n-1,L}$
7: \hspace{2em} $\{Q_i, h_n(\cdot)\} = \text{UPDATE}([t_i, t_{i+1}], \phi_{n-1}(\cdot), h_n(t_i))$
8: \hspace{1em} $L^n = \text{MERGE}(L^n \cup Q_i)$
9: end for
10: end for

We remark that $h_n(z)$ admit the unified expression as in (23) for $z \in [B, u_b]$. The proof in the above lemma also shows a way to update $h_n$, which is summarised in Algorithms 2 and 3.

The final main issue is that once $h_n(y_n)$ is constant in two consecutive intervals after the Update procedure, we need to merge the two intervals so that the later backtracking procedure, which is used to recover the optimal solution $y^*$, is well defined. Specifically, suppose that after the Update procedure, $h_n(y_n)$ is constant in two consecutive intervals $[s_{i-1}^n, s_i^n]$ and $[s_i^n, s_{i+1}^n]$. Note that due to continuity, the constant values of $h_n$ in the two intervals are the same. Then we merge the two intervals into $[s_{i-1}^n, s_{i+1}^n]$ and obtain that $h_n$ is constant in $[s_{i-1}^n, s_{i+1}^n]$.

**Definition 4** (Merge). The Merge procedure merges all consecutive intervals where $h_n$ is constant. It works as $S = \text{MERGE}(T)$, where $T$ consists of consecutive intervals, and $S$ is a single interval.

**Remark 2.** We remark that when $h_n$ is not constant in two consecutive pieces, the case that $h_n$ admits the same coefficients cannot occur. Indeed, by our update rule, in each piece $h_n$ is determined by a constant generated by $h_k$ and $\sum_{i=k}^{n-1} f_i(\cdot)$ where $k \leq n - 1$. If the two consecutive pieces share the same $\sum_{i=k}^{n-1} f_i(\cdot)$, then the constants must be the same because of the continuity of $h_n$. This contradicts our Merge step as we have merged the consecutive constant intervals in $h_k$ for $k < n$.

Now we are ready to summarise the update of $h_n$ based on the knowledge of $h_{n-1}$ in $[l_b, B]$ in Algorithm 2. We remark that as we have derived the lower bound $l_b$ in Lemma 4 for $y^*$, it must holds that $s_1^{n-1} = l_b$ and $s_{K_{n-1}^R+1}^{n-1} = B$.

We also describe an analogous approach for updating $h_n$ in the interval $[B, u_b]$ in Algorithm 3. The only difference is that at the beginning $h_n(r_1^{n-1}) = h_n(B)$ is already computed when updating $h_n$ in $[l_b, B]$. Thus we do not need line 1 in Algorithm 2. Just like the case in $[l_b, B]$, $h_{n-1}(\cdot)$ is a piecewise function with $K_{n-1}^R$ intervals in $[B, u_b]$. In each interval $[r_k^{n-1}, r_{k+1}^{n-1}]$, $h_n(z)$ still have the expression (23) and the recursion to update $h_n$ still work. So all the procedures for updating $h_n$ are the same as those in the case of $[l_b, B]$.

To introduce the full DP algorithm, we give more properties on the Merge procedure. Due to the special structure of $f_{n-1}(\cdot)$, the piecewise expression of $h_n(\cdot)$ is naturally split by $[l_b, B]$ and
Algorithm 3 The deduction of \( h_n(\cdot) \) from \( h_{n-1}(\cdot) \) in \([B, ub]\)

Input: \( h_{n-1}(\cdot), f_{n-1}(\cdot), h_n(B) \), and \( R^{n-1} = \{ [r_{n-1}^{n-1}, r_{n-2}^{n-1}], [r_{n-2}^{n-1}, r_{n-3}^{n-1}], \ldots, [r_{K_{n-1}}^{n-1}, r_{K_{n-1}+1}^{n-1}] \} \) with 
\( r_{1}^{n-1} = B \) and \( r_{K_{n-1}+1}^{n-1} = ub \)

Output: \( \{ h_n(\cdot), R^n \} \)

1: \( R^n = \emptyset \)
2: for \( k=1, 2, \ldots, K_{n-1} \) do 
3: \( T_k^{n-1, R} = \text{DECOMPOSE}(r_{k}^{n-1}, r_{k+1}^{n-1}, \phi_{n-1}(\cdot)) \)
4: \( \text{for } i = 1, \ldots, \text{Card}(T_k^{n-1, R}) \) do 
5: \( \text{select } [t_i, t_{i+1}] \in T_k^{n-1, R} \)
6: \( \{ Q, h_n(\cdot) \} = \text{UPDATE}([t_i, t_{i+1}], \phi_{n-1}(\cdot), h_n(t_i)) \)
7: \( R^n = \text{MERGE}(R^n \cup Q) \)
8: end for
9: end for

\([B, ub]\). So in the forward deduction of \( h_n(\cdot) \), we adopt \text{MERGE} step in \([l_b, B]\) and \([B, ub]\) separately.

It is possible that \( h_n \) admits the same constant in the first interval \([s_{K_{n}^L}, B]\) in \( L_n \) and the last interval \([B, r_{n}^{n}]\) in \( R_n \). But in this case, in the backward procedure to obtain the optimal solution \( \phi(y) \) using (19), which will be introduced in the following paragraphs, we need to merge \([s_{K_{n}^L}, B]\) and \([B, r_{n}^{n}]\).

**Definition 5 (MidMerge).** We refer to the above mentioned procedure of merging consecutive intervals where \( h_n \) is constant covering \( B \) as \text{MidMerge}, i.e., \( H_n = \text{MidMerge}(L^n, R^n) \).

It is easy to see the cardinality of \( H_n \) is either \( K_n = K_{n}^L + K_{n}^R - 1 \) or \( K_n = K_{n}^L + K_{n}^R \), depending on if the last piece in \( L_n \) and the first piece in \( R_n \) are merged. We represent \( H_n \) as \( H_n = \{ [s_{k}^{n}, s_{k+1}^{n}], k = 1, 2, \ldots, K_n \} \).

Now we have been able to deduce the explicit formula for every \( h_n(\cdot), n = 1, 2, \ldots, N + 1 \). We still need a backward procedure to obtain the optimal value for \( y_N^*, y_{N-1}^*, \ldots, y_2^*, y_1^* \). To this end, we introduce several lemmas below.

**Lemma 7.** If \( h_n(z) \) is strictly decreasing in \([s_{k}^{n}, s_{k+1}^{n}]\), then \( h_n(z) = \phi_{n-1}(z) \) for \( z \in [s_{k}^{n}, s_{k+1}^{n}] \).

**Proof.** The proof follows directly from (21).

**Lemma 8.** Consider the stage after \text{Merge} and \text{MidMerge}. Suppose that \( h_n(z) \) is a constant in \([s_{k}^{n}, s_{k+1}^{n}]\). Then \( h_n(z) = \phi_{n-1}(s_{k}^{n}) \) for all \( z \in [s_{k}^{n}, s_{k+1}^{n}] \), and thus \( \phi_{n-1}(y) \geq \phi_{n-1}(s_{k}^{n}) \) for all \( y \in [s_{k}^{n}, s_{k+1}^{n}] \).

**Proof.** As we merged consecutive intervals that \( h_n \) are identical constants, we see that the fact that \( h_n(y_n) \) is a constant in \([s_{k}^{n}, s_{k+1}^{n}]\) implies that in the previous interval \([s_{k-1}^{n}, s_{k}^{n}]\), \( h_n(y_n) \) is strictly decreasing. From Lemma 7, we have \( h_n(s_{k}^{n}) = \phi_{n-1}(s_{k}^{n}) \). The continuity of \( h_n(\cdot) \) further guarantees that \( h_n(z) = h_n(s_{k}^{n}) = \phi_{n-1}(s_{k}^{n}) \) for all \( z \in [s_{k}^{n}, s_{k+1}^{n}] \). From (22), we immediately have that \( \min_{k \leq y \leq s_{k+1}^{n}} \phi_{n-1}(y) \geq \phi_{n-1}(s_{k}^{n}) \) for all \( y \in [s_{k}^{n}, s_{k+1}^{n}] \), and thus \( \phi_{n-1}(y) \geq \phi_{n-1}(s_{k}^{n}) \) for all \( y \in [s_{k}^{n}, s_{k+1}^{n}] \).
Now we are ready to present the backward recursion for obtaining the optimal solution \( y^* \). The original problem of (14) is equivalent to (18). Using (19), we can recursively update \( y^*_n \) backward, by setting \( y^*_{N+1} = u_b \). Now we will show the details of how to find \( y^*_n \) using \( y^*_n \) for \( n = N+1, N, \ldots, 2 \). Suppose \([s^n_k, s^n_{k+1}]\) is the piece of \( h_n \) that contains \( y^*_n \). We update \( y^*_{n-1} \) according to the following two cases.

1. \( h_n(y) \) is strictly decreasing in the interval \([s^n_k, s^n_{k+1}]\). Then \( h_n(y) = \phi_{n-1}(y), y \in [s^n_k, s^n_{k+1}] \) due to Lemma 7. So \( \phi_{n-1}(y) \) is strictly decreasing in \([s^n_k, s^n_{k+1}]\). Particularly, \( \phi_{n-1}(s^n_k) = h_n(s^n_k) \). Therefore \( h_n(s^n_k) \geq \phi_{n-1}(y), \forall y \in [s^n_k, y^*_n] \). From (21), we have

\[
h_n(s^n_k) = \min_{x \leq s^n_k} \phi_{n-1}(x) \leq \phi_{n-1}(z), \quad \forall z \leq s^n_k.
\]

The above two facts imply \( \phi_{n-1}(z) \geq \phi_{n-1}(y) \) for all \( z \leq s^n_k \) and \( y \in [s^n_k, y^*_n] \) and thus

\[
y^*_{n-1} \overset{(19)}{=} \text{argmin}_{y \leq y^*_n} \phi_{n-1}(y) = \text{argmin}_{s^n_k \leq y \leq y^*_n} \phi_{n-1}(y) = y^*_n.
\]

The last equations is because \( \phi_{n-1}(y) \) is strictly decreasing in \([s^n_k, y^*_n]\).

2. \( h_n(y) \) is constant in the interval \([s^n_k, y^*_n]\). This means \( \phi_{n-1}(y) \geq h_n(s^n_k) = \phi_{n-1}(s^n_k), y \in [s^n_k, y^*_n] \), where the second equality is due to Lemma 8. This, together with (21), guarantees that \( \phi_{n-1}(s^n_k) = h_n(s^n_k) \leq \phi_{n-1}(y), \forall y \leq s^n_k \). Therefore as \( h_n(y) \) is a constant in the interval \([s^n_k, y^*_n]\), we have

\[
\phi_{n-1}(s^n_k) = h_n(s^n_k) = h_n(y^*_n) = \min_{y_{n-1} \leq y^*_n} \phi_{n-1}(y_{n-1}),
\]

which is equivalent to

\[
y^*_n = \text{argmin}_{y_{n-1} \leq y^*_n} \phi_{n-1}(y_{n-1}) = s^n_k.
\]

We summarize the full DP algorithm in Algorithm 4. We remark that in line 6, we introduce the dummy variable \( y^*_N = u_b \) according to (18). The following theorem is then straightforward.

**Theorem 2.** Algorithm 4 correctly returns a global optimal solution for problem (12).

The time complexity of our DP algorithm mainly depends on the number of pieces of intervals in each \( h_n \). Assume this number is \( O(M) \). Recall that each DECOMPOSE and UPDATE step invokes at most twice the root-finding oracle. As there are \( N \) stages in the forward deduction, the DP algorithm invokes the oracle at most \( O(MN) \) times. And in the backward recursion, at most \( O(MN) \) intervals need to be traversed, which is faster than the forward stage as no oracle needs to be invoked. Therefore the DP algorithm invokes the oracle \( O(MN) \) times. In the worst case, an interval may generate four new intervals in each iteration after the DECOMPOSE and UPDATE step. This gives an upper bound for \( M \), i.e., \( M \leq 4^N \). However, in our (thousands of) numerical experiments, \( M \) is always around \( N \). Typically, \( h_{N+1} \) contains the most intervals than \( h_i, i = 1, \ldots, N \), from our empirical observation. For an illustration, we show the interval number against \( N \) in Figure 2 for an instance of Fama French 48 Industries in Section 5.2. In practice, we may roughly think that the oracle complexity of our DP method is \( O(N^2) \).
Algorithm 4 A dynamic programming algorithm for solving (14)

Input: $f_i, i = 1, \ldots, N$, $h_1(z) = 0$, $L^1 = \{[l_b, B]\}$, $R^1 = \{[B, u_b]\}$

Output: $\{y^*_1, y^*_2, \ldots, y^*_N\}$

1: for $n = 2, 3, \ldots, N + 1$ do
2:   $\{h_n(\cdot), L^n\}$ returned by Algorithm 2
3:   $\{h_n(\cdot), R^n\}$ returned by Algorithm 3
4: end for
5: $H_{N + 1} = \text{MidMerge}(L_{N + 1}, R_{N + 1})$
6: $y^*_{N + 1} = u_b$
7: for $n = N + 1, N, \ldots, 2$ do
8:   $H_n = \text{MidMerge}(L_n, R_n)$
9:   Find $[s^n_k, s^n_{k + 1}] \in H_n$, such that $y^*_n \in [s^n_k, s^n_{k + 1}]$
10: if $h_n(\cdot)$ is a constant in $[s^n_k, s^n_{k + 1}]$ then
11:   $y^*_{n - 1} = s^n_k$
12: else
13:   $y^*_{n - 1} = y^*_n$
14: end if
15: end for

4.2 Two-partition Pool-adjacent-violators Algorithm

The DP method returns a global optimal solution for the $y$-subproblem, but it suffers from a relatively low speed. Alternatively, we propose a much faster algorithm for the $y$-subproblem, which computes a local minimizer for the $y$-subproblem (5b).

The Pooling Adjacent Violators (PAV) algorithm is one of the most successful algorithms in solving separable convex optimization subject to the chain constraint. The PAV algorithm was first proposed for maximum likelihood estimation (MLE) under the chain constraint in Ayer et al. (1955) and Brunk et al. (1972). Best and Chakravarti (1990) theoretically explained the PAV algorithm as a dual active set algorithm when the separable objective function is a $l_2$ norm. The PAV algorithm was generalized to solving separable convex objective functions in Strömberg (1991) and Best et al. (2000). Moreover, when an integer solution is required, a fast scaling PAV algorithm was proposed in Ahuja and Orlin (2001). Recently, a variant of the PAV algorithm for solving a special class of non-convex isotonic regression was proposed in Cui et al. (2021). Unfortunately, the method in Cui et al. (2021) cannot be applied to solve our problem as it requires the objective function in a special form that is completely different from problem (14). Despite the non-convexity of both (5b) and the reformulation (14), we propose an algorithm based on the PAV algorithm using the special structure of $f_i(y)$.

Now let us describe our variant of the PAV algorithm. Its basic idea is that using the special structure of problem (14), we first split the variables into two consecutive intervals, and then use two classical PAV algorithms to solve the two problems separately. The key observation is that each PAV algorithm returns a solution satisfying the chain constraint and remaining in its original interval. Hence combining the two pieces of solutions forms a feasible solution for the problem (14), which is indeed a local minimizer as to be shown in Theorem 3. The main feature of our algorithm is the partition of variables into two intervals and the usage of the PAV algorithm. Hence we term our
algorithm two-partition pool-adjacent-violators (TPPAV). Our numerical result suggests that the proposed algorithm has a good practical performance.

Now let us first recall the traditional PAV algorithm, which aims to solve the following problem

$$\min_{1 \leq i \leq n} \theta_i(y_i), \quad \text{s.t. } y_1 \leq y_2 \leq \cdots \leq y_n,$$

where each $\theta_i$ is a univariate convex function. The PAV algorithm maintains a set $J$ that partitions indices $\{1, 2, \ldots, n\}$ into consecutive blocks $[s_1, s_1 + 1], [s_2, s_2 + 1], \ldots, [s_t, n]$, where $[a, b]$ denotes the index set $\{a, a + 1, \ldots, b\}$ for positive integers $a < b$, and we define by convention $[a, a] = \{a\}$. A block $[p, q]$ is a single-valued block if the coordinate of the optimal solution of the following problem has the same value,

$$\min_{p \leq i \leq q} f_i(y_i), \quad \text{s.t. } y_p \leq y_{p+1} \leq \cdots \leq y_q,$$

i.e., $y_p^* = y_{p+1}^* = \cdots = y_q^*$, where in the PAV literature $V_{[p,q]}$ is used to denote this value. For two consecutive blocks $[p, q]$ and $[q + 1, r]$, if $V_{[p,q]} \leq V_{[q+1,r]}$ then the two blocks are in-order, otherwise out-of-order. The PAV algorithm initially partitions every integer from 1 to $n$ as single-valued blocks $[i, i], i = 1, \ldots, n$. Once there exists consecutive out-of-order single-valued blocks $[p, q]$ and $[q + 1, r]$, the PAV algorithm merges these two blocks by replacing them with the larger block $[p, r]$. When all the single-valued blocks are in-order, the PAV algorithm terminates.

The TPPAV algorithm is motivated by the observation that $f_i(\cdot)$ is convex in either $[B, u_b]$, or $[t_b, B - \rho_i]$, where $\rho_i = \left(\frac{\mu_0(1-\alpha)\sigma}{\sigma}\right)^{\frac{1}{\gamma-1}}$. Here the convexity in $[B, u_b]$ and $[t_b, B - \rho_i]$ follows directly from Propositions 1 and 2, respectively. In general, using the convention

$$h_{[p,q]}(y) = \sum_{i=p}^{q} f_i(y),$$

we show in the following lemma that such properties also hold for $h_{[p,q]}(y)$. 

![Figure 2: Interval number of $h_{N+1}$](image)
Lemma 9. The function $h_{[p,q]}(y)$ is strongly convex in either $(-\infty, B - \rho_{[p,q]})$ or $(B, \infty)$, where $\rho_{[p,q]} = \max_{i=p+1,\ldots,q} \rho_i$. Moreover, $h_{[p,q]}(y)$ has at most one local minimizer in $(-\infty, B - \rho_{[p,q]})$, and exactly one local minimizer in $(B, \infty)$.

Proof. The formula of $h_{[p,q]}(y)$ is indeed

$$h_{[p,q]}(y) = \begin{cases} \sum_{i=p}^{q} a_i(B - y)^\alpha + \sigma 2(y - w_i)^2, & y \leq B, \\ \sum_{i=p}^{q} -b_i(y - B)^\alpha + \sigma 2(y - w_i)^2, & y > B, \end{cases}$$

which is equivalent to

$$h_{[p,q]}(y) = \begin{cases} (q - p + 1)(\bar{a}_{[p,q]}(B - y)^\alpha + \sigma 2(y - \bar{w}_{[p,q]})^2) + M_{[p,q]}, & y \leq B, \\ (q - p + 1)(\bar{b}_{[p,q]}(y - B)^\alpha + \sigma 2(y - \bar{w}_{[p,q]})^2) + M_{[p,q]}, & y > B, \end{cases}$$

where $\bar{a}_{[p,q]}$ is the mean of $\{a_p, a_{p+1}, \ldots, a_q\}$, $\bar{b}_{[p,q]}$ is the mean of $\{b_p, b_{p+1}, \ldots, b_q\}$, $\bar{w}_{[p,q]}$ is the mean of $\{w_p, w_{p+1}, \ldots, w_q\}$, and $M_{[p,q]} = \frac{\sigma}{2} \left( \sum_{i=p}^{q} w_i^2 \right) - (q - p + 1)\bar{w}_{[p,q]}$ is a constant.

Note that $h_{[p,q]}(y)$ can be seen as $\zeta(y)$ plus a constant. As $\bar{a}_{[p,q]} \leq \max_{i=p} a_i$, we obtain that $h_{[p,q]}(y)$ is convex in $(-\infty, B - \rho_{[p,q]})$.

Since $\zeta(y)$ is convex in $(B, \infty)$, we obtain that $h_{[p,q]}(y)$ is convex in $(B, \infty)$. \hfill \Box

Based on Lemma 9, we partition the indices into two sets. Let $K$ be the smallest $i$ such that $f_i(y)$ either has no local minimizer in the interval $(-\infty, B)$ or has a local minimizer $\tau_i$ with $\tau_i \geq B - \rho_{[1,K]}$. For national simplicity, we define $\delta = \rho_{[1,K-1]}$ and two index sets

$$J_1 = \{1, 2, \ldots, K - 1\} \text{ and } J_2 = \{K, K + 1, \ldots, N\}.$$

The above definition means that for all $i < K$, $f_i(y)$ has exactly one local minimizer in $(-\infty, B - \delta]$, and for all $i \geq K$, $f_i(y)$ has exactly one local minimizer in $[B, \infty)$. For $h_{[p,q]}$ with $p, q \in J_1$, we compute the local minimizer in $[b, B - \delta]$, which is the left 0-derivative point of $h_{[p,q]}$; instead of the global minimizer in the original PAV algorithm. For $i \in J_2$, we similarly use the local minimizer in $[B, u_b]$, which is the only 0-derivative point of $h_{[p,q]}$, instead of the global minimizer in the original PAV algorithm.

The well-definedness of the existence of a local minimizer are guaranteed by the following proposition.

Proposition 3. Suppose $V_{[m,n]} > V_{[n+1,p]}$. Then there exists a local minimizer, denoted by $V_{[m,p]}$, for $h_{[m,p]}(y)$ in the interval $(V_{[n+1,p]}, V_{[m,n]})$, i.e., the procedure in lines 5-9 (pooling adjacent violator step) in Algorithm 5 is well defined. Moreover, $V_{[m,p]}$ satisfies

$$V_{[n+1,p]} < V_{[m,p]} < V_{[m,n]}.$$

Proof. We first consider the restriction to $J_1$. Due to Lemma 9, we have

$$h'_{[m,n]}(V_{[m,n]}) = 0 \text{ and } h'_{[n+1,p]}(V_{[n+1,p]}) = 0.$$
Therefore we obtain that

\[ y \text{ encourages generating a solution with more components of } \mathbf{y} \text{ than the reference point } B. \]

We summarise our algorithm in Algorithm 5. We remark that our selection of \( K \) in the partition encourages generating a solution with more components of \( \mathbf{y} \) greater than the reference point \( B \). This is consistent with our goal to maximize utility.

Finally, we show in the next theorem that TPPAV generates a strictly local minimizer for the \( \mathbf{y} \)-subproblem of the proposed ADMM algorithm.

**Theorem 3.** The solution \( \mathbf{y} \) returned by the TPPAV algorithm is a strictly local minimizer of problem (14). Thus it holds that \( \mathbf{0} \in \partial \Phi(\mathbf{y}) \).

---

**Algorithm 5.** A Two-partition Pool-adjacent-violators algorithm for solving (14)

**Input:** \( J_1 = \{[1,1], [2,2], \ldots, [K-1,K-1]\} \), \( J_2 = \{[K,K], [K+1,K+1], \ldots, [N,N]\} \), \( I_1 = [l_b,B-\delta] \) and \( I_2 = [B,u_b] \)

**Output:** \( \{y_1,y_2,\ldots,y_N\} \)

1. for \( k = 1,2 \) do
2.   for each \([i,i] \in J_k\) do
3.     compute the local minimizer \( V_{[i,i]} \) of \( f_i(y) \) in \( I_k \).
4.   end for
5. while \( \exists [m,n],[n+1,p] \in J_k \) such that \( V_{[m,n]} > V_{[n+1,p]} \) do
6.   \( J_k \leftarrow J_k \setminus \{[m,n],[n+1,p]\} \cup \{[m,p]\} \)
7.   \( h_{[m,p]}(y) \leftarrow \sum_{i=m}^p f_i(y) \)
8.   compute the local minimizer \( V_{[m,p]} \) of \( h_{[m,p]}(y) \) in \( I_k \)
9. end while
10. for each \([m,n] \in J_k\) do
11.   \( y_i = V_{[m,n]}, \forall i = m,m+1,\ldots,n \)
12. end for
13. end for

Noting that \( B - \delta > V_{[m,n]} > V_{[n+1,p]} \), we further have

\[ h'_{[m,n]}(V_{[n+1,p]}) < 0, \quad \text{and} \quad h'_{[n+1,p]}(V_{[m,n]}) > 0. \]

Therefore we obtain that

\[ h'_{[m,p]}(V_{[n+1,p]}) = h'_{[m,n]}(V_{[m,n]}) + h'_{[n+1,p]}(V_{[m,n]}) > 0, \]
\[ h'_{[m,p]}(V_{[n+1,p]}) = h'_{[m,n]}(V_{[n+1,p]}) + h'_{[n+1,p]}(V_{[n+1,p]}) < 0. \]

As \( h_{[m,p]} \) is strongly convex in \((V_{[n+1,p]},V_{[m,n]})) \subset (-\infty,B-\delta)\) due to Lemma 9, the first order derivative \( h'_{[m,p]} \) is strictly increasing in \((V_{[n+1,p]},V_{[m,n]}))\). The above facts, together with the strong convexity of \( h_{[m,p]} \) in \((\infty,B-\delta)\) and the fact that \( h_{[m,p]} \) is in the form of \( \zeta \), imply that there exists one local minimizer of \( h'_{[m,p]} \) in \((V_{[n+1,p]},V_{[m,n]}))\), which is exactly \( V_{[m,p]} \). This further implies the desired property.

The case of \( J_2 \) is similar to the previous case, and its proof is omitted for simplicity. \( \Box \)
Proof. Let the final blocks of TPPAV be as $\{[s_1, s_2], [s_2 + 1, s_3], \ldots, [s_k + 1, s_{k+1}]\}$, where $s_1 = 1$ and $s_{k+1} = N$. Suppose $y_{ji} = y_{i[q]}$ for $i = 1, \ldots, N$. Denote by $a_{i:j}$ the vectors formed by the $i$th to $j$th entries of vector $a$. From Lemma 9 and Algorithm 5, we obtain that $\bar{y}_{j_{t-1}, j_{t+1}}$ is a strictly local minimizer of
\[
\min_{s_1 \leq t \leq s_k} \sum_{i=s_t}^{s_{t+1}} f_{j_i}(y_{ji})
\]
for $t = 1 : k$. Then we have obtained that $\bar{y}$ is a strictly local minimizer of $\min_{i=1}^{N} f_i(y_{i[q]})$. Because $\bar{y}$ satisfies the chain constraint, $\bar{y}$ is a strictly local minimizer of (14).

Theorem 4. Given a root finding oracle for computing local minimizers in lines 3 and 8, Algorithm 5 invokes the root-finding oracle at most $2N - 1$ times.

Proof. First note that line 3 invokes the oracle $N$ times. Next consider the case that the first partition $J_1$ is nonempty. Then $J_1$ contains at most $K - 1$ blocks at the beginning, and we need to find the local minimizer less than $B - \delta$ for these $K - 1$ blocks. In the pooling adjacent violators period (lines 5-9 in Algorithm 5), once we pool two blocks, the number of blocks in $J_1$ is reduced by 1. Therefore we need at most $K - 1$ merging steps. So line 8 invokes the oracle $N - 1$ times in the case that $J_2$ is empty and $N - 2$ otherwise. Similarly, this holds for the case that $J_1$ is empty. In summary, Algorithm 5 invokes the oracle at most $2N - 1$ times.

We finally give a remark on the runtime of a function evaluation of each oracle in line 8. In each merging step, the time complexity of calculating the block objective function $h_{[p,r]}$ is $O(1)$ due to the special structure of $h_{[p,r]}$. Indeed, in the merging step, deriving the coefficients of $h_{[p,r]}$ from $h_{[p,q]}, h_{[q+1,r]}$ can be done $O(1)$ time since
\[
a_{[p,r]} = \frac{(q-p+1)a_{[p,q]} + (r-q)a_{[q+1,r]}}{r-p+1}
\]
and $\tilde{b}_{[p,r]}, \tilde{w}_{[p,r]}$ can be computed similarly. Once the coefficients in $h_{[p,r]}$ are calculated, the root-finding oracle can be used to find a local minimizer of $h_{[p,r]}$. We should point out that the previous PAV algorithm in Ahuja and Orlin (2001); Strömberg (1991); Best et al. (2000) requires $O(N)$ time to evaluate a general block objective function as it involves $O(N)$ additions of $f_i$, which is more expensive than $O(1)$ in our algorithm.

4.3 A Hybrid algorithm

A disadvantage of the TPPAV algorithms is that the solution may not satisfy the descent property (9) required by the ADMM algorithm, though it is much faster than the DP algorithm in both theory and practice. The non-descent cases are observed in our empirical experiments. To remedy this, we propose a hybrid algorithm. It first uses the TPPAV algorithm to get an approximate solution, and then accepts this solution if it satisfies the descent property (9). Otherwise it invokes the DP algorithm to get a global solution to obtain the decrease in (9). Recalling that $\Phi(y) = \sum_{i=1}^{N} f_i(y_i)$ is the objective for $y$-subproblem, we summarize this procedure in Algorithm 6.


**Algorithm 6** A Hybrid algorithm for solving (14)

**Input:** $\Phi(\cdot)$ and $y^k$

**Output:** $y^{k+1}$

1. Use Algorithm 5 (the TPPAV algorithm) to obtain $y^{k+1}$
2. if $\Phi(y^{k+1}) > \Phi(y^k)$ then
3. Use Algorithm 4 (the DP algorithm) to obtain a new $y^{k+1}$
4. end if

5 Numerical Experiments

We test the performance of the proposed methods in this section. We first compare the performance of our two $y$-subproblem solvers: the DP and TPPAV algorithms. Then we compare the ADMM algorithms with three different $y$-subproblem solvers: the DP algorithm, the TPPAV algorithm, and the hybrid algorithm. All the experiments are implemented using MATLAB R2021b on a PC running Windows 10 Intel(R) Xeon(R) E5-2650 v4 CPU (2.2GHz) and 64GB RAM.

5.1 Numerical Tests for the $y$-subproblem Solvers

In this subsection, we aim to test whether our proposed two $y$-subproblem is capable of numerical efficiency. We compare the average performance for the DP and TPPAV algorithms and the `fmincon` solver in MATLAB.

We test different instances with scenario number $N = 50, 100, 200, 300, 400, 500$. We generate 10 instances for each scenario number, where every component of $w$ in (12) is uniformly generated in $[-0.1, 0.1]$.

| Scenarios | DP ($\times 10^{-2}$) | TPPAV ($\times 10^{-2}$) | fmincon ($\times 10^{-2}$) |
|-----------|-----------------------|--------------------------|---------------------------|
| $N = 50$  | 4.3058                | 4.3058                   | 4.3126                    |
| $N = 100$ | 4.6096                | 4.6096                   | 4.6101                    |
| $N = 200$ | 4.8137                | 4.8137                   | 4.8227                    |
| $N = 300$ | 4.5241                | 4.5241                   | 4.5384                    |
| $N = 400$ | 4.8163                | 4.8163                   | 4.8189                    |
| $N = 500$ | 4.9135                | 4.9135                   | 4.9138                    |
Table 2: Time comparison (seconds) for the y-subproblem (12).

| Scenarios | DP      | TPPAV   | fmincon |
|-----------|---------|---------|---------|
| $N = 50$  | 0.6469  | 0.0126  | 0.6053  |
| $N = 100$ | 2.49    | 0.0122  | 1.35    |
| $N = 200$ | 10.9    | 0.0298  | 3.371   |
| $N = 300$ | 27.25   | 0.0427  | 6.763   |
| $N = 400$ | 53.27   | 0.0701  | 11.57   |
| $N = 500$ | 92.68   | 0.0938  | 19.84   |

Table 1 reports the comparison of averaged objective values for different methods. It shows that both the DP and TPPAV algorithms find better (smaller) objective values than fmincon for different scenario numbers. Table 2 reports the time comparison for different methods. It shows that the TPPAV algorithm is the fastest method among the three and is much faster than the second fastest solver fmincon. This verifies our complexity result that shows TPPAV only invokes at most $2N - 1$ root-finding oracles. Table 1 shows that both the DP and TPPAV algorithms always obtain the same objective values. As the DP algorithm is guaranteed to obtain the global minimum, we see that in the tested random instances, the TPPAV algorithm indeed achieves the global minimum. However, we find that in real data experiments to be presented in the next subsection this is not always true. The quality of the TPPAV solution depends on our heuristic choice of $K$. We checked one specific instance. In the optimal solution, there is 13 components smaller than the reference point $B$, whereas $K = 14$ was set in our TPPAV algorithm. This leads to an increase in the objective value, which causes a violation to the descent condition (9) in our Assumption 2. Such a drawback can be avoided by the hybrid algorithm.

5.2 Numerical Tests for the ADMM Algorithms on Real Data

In this subsection, we compare the performance of our ADMM algorithm based on three subproblem solvers in Section 4 and the fmincon solver in MATLAB for solving the CPT portfolio optimization problem.

The numerical experiment data is drawn from real historical returns of Fama French 48 Industries (FF48) with 48 assets[^1] and Standard & Poor’s 500 with 458 assets[^2]. We adopt a different number of scenarios in generating the historical return $R$. We test $N = 50, 100, 150, 200, 250,$ and $300$ days of historical return $R$ to have an all-round assessment of our three algorithms for FF48. For S&P 500, we test $N = 300, 400, 500, 600, 700, 800, 900, 1000$. The FF48 dataset and the S&P 500 dataset range from December 2016 to December 2017 and August 2009 to August 2013, respectively. Note that as the component stocks of S&P 500 may change over time, we choose the period August 2009 to August 2013 during which S&P 500 has the same components.

We use typical stopping criteria in ADMM (Boyd et al., 2011), where $\|y^k - Rx^k\| < \epsilon_1$ and $\|y^k - y^{k-1}\| < \epsilon_2$ is referred to as primal and dual feasibility, respectively. To better facilitate the

[^1]: Downloaded from Kenneth R. French’s website: [http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library](http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library)

[^2]: Downloaded from Wharton Research Data Services [https://wrds-www.wharton.upenn.edu/](https://wrds-www.wharton.upenn.edu/).
convergence, we dynamically update the penalty parameter $\sigma$ after updating $\lambda$ in the following way

$$
\sigma^{k+1} = \begin{cases} 
\omega \sigma^k, & \text{if } \|y^{k+1} - Rx^{k+1}\| > \epsilon_1, \\
\sigma^k, & \text{otherwise,}
\end{cases}
$$

where $\epsilon > 1$ is a constant. In all the three ADMM algorithms, we set $\sigma^0 = 10$, $\omega = 3$, $\epsilon_1 = 5 \times 10^{-5}$, and $\epsilon_2 = 2 \times 10^{-5}$ for FF48 and $\epsilon_2 = 10^{-4}$ for S&P 500. We set maximal iteration number as 1000, maximal time as one hour, and initialize $y^0 = \lambda^0 = 0$.

In all three ADMM algorithms, the $x$-subproblem was solved via the off-the-shelf solver Gurobi 9.5.1 (Gurobi Optimization, LLC, 2022). The $y$-subproblem was solved using the three proposed subproblem solvers in Section 4. We term the three ADMM algorithms as ADMM-DP, ADMM-TPPAV, and ADMM-Hybrid when using the DP, TPPAV, and hybrid algorithms for solving the $y$-subproblem. We use the $\text{fmincon}$ solver to directly solve problem (1) with default setting\(^3\). Note that problem (1) is non-convex and non-smooth. Therefore there is no theoretical guarantee of the solution returned by $\text{fmincon}$.

| Assets | Scenarios | ADMM-DP     | ADMM-TPPAV  | ADMM-Hybrid | $\text{fmincon}$ |
|--------|-----------|-------------|-------------|-------------|-----------------|
| $d=48$ | $N=50$    | $-1.854 \times 10^{-3}$ | $-1.369 \times 10^{-3}$ | $-1.369 \times 10^{-3}$ | $-8.754 \times 10^{-4}$ |
| $d=48$ | $N=100$   | $-3.478 \times 10^{-4}$  | $-3.529 \times 10^{-4}$ | $-3.424 \times 10^{-4}$ | $1.981 \times 10^{-4}$ |
| $d=48$ | $N=150$   | $4.411 \times 10^{-4}$  | $4.463 \times 10^{-4}$ | $4.407 \times 10^{-4}$ | $1.495 \times 10^{-3}$ |
| $d=48$ | $N=200$   | $6.385 \times 10^{-4}$  | $6.388 \times 10^{-4}$ | $6.386 \times 10^{-4}$ | $9.839 \times 10^{-4}$ |
| $d=48$ | $N=250$   | $1.195 \times 10^{-3}$ | $1.200 \times 10^{-3}$ | $1.195 \times 10^{-3}$ | $1.461 \times 10^{-3}$ |
| $d=48$ | $N=300$   | $2.323 \times 10^{-3}$* | $2.326 \times 10^{-3}$ | $2.323 \times 10^{-3}$ | $2.453 \times 10^{-3}$ |
| $d=458$| $N=300$   | $3.4337 \times 10^{-3}$ | $3.3782 \times 10^{-3}$ | $3.3783 \times 10^{-3}$ | $7.0031 \times 10^{-3}$ |
| $d=458$| $N=400$   | $3.2268 \times 10^{-3}$ | $3.2258 \times 10^{-3}$ | $3.2257 \times 10^{-3}$ | $6.7612 \times 10^{-3}$ |
| $d=458$| $N=500$   | $4.6362 \times 10^{-3}$ | $4.6563 \times 10^{-3}$ | $4.6566 \times 10^{-3}$ | $8.5395 \times 10^{-3}$ |
| $d=458$| $N=600$   | $5.0974 \times 10^{-3}$ | $5.0304 \times 10^{-3}$ | $5.0311 \times 10^{-3}$ | $8.9193 \times 10^{-3}$ |
| $d=458$| $N=700$   | $4.9923 \times 10^{-3}$ | $4.8813 \times 10^{-3}$ | $4.8813 \times 10^{-3}$ | $8.6625 \times 10^{-3}$ |
| $d=458$| $N=800$   | $5.1860 \times 10^{-3}$ | $4.9342 \times 10^{-3}$ | $4.9342 \times 10^{-3}$ | $8.5305 \times 10^{-3}$ |
| $d=458$| $N=900$   | $5.1305 \times 10^{-3}$ | $4.7083 \times 10^{-3}$ | $4.7081 \times 10^{-3}$ | $8.1760 \times 10^{-3}$ |
| $d=458$| $N=1000$  | $5.8797 \times 10^{-3}$ | $5.1949 \times 10^{-3}$ | $5.1949 \times 10^{-3}$ | $8.3533 \times 10^{-3}$ |

\(^{3}\)We do not provide gradient for $\text{fmincon}$ because the gradient of the objective function in (1) with regard to $x$ is unavailable due to the non-smoothness of the S-shape utility.
We compare final objective values and CPU time among different algorithms. The objective value is the negative of the CPT utility function, where a lower objective value means a better solution. In all the ADMM algorithms, the objective values are calculated directly from $x$ instead of using the intermediate variable $y$. We report the comparisons on objective values in Table 3 and the CPU time in Table 4, respectively. In Table 3 we include the objective value of fmincon to prove the solution quality of our ADMM algorithms. In both tables, “∗” indicates an instance that ADMM-DP fails to converge in one hour, and “−” indicates that the ADMM-Hybrid algorithm does not invoke the DP algorithm for the $y$ subproblem.

From Table 3 we find that all the three ADMM algorithms outperformed the default fmincon in final objective values. On the other hand, we observe that the ADMM-DP is time-consuming from Table 4, where the ADMM-DP algorithm always reached the time limit for large $N$. Indeed, the ADMM-DP failed to meet the convergence tolerance in one hour when $N ≥ 500$ because the DP algorithm for solving the $y$-subproblem is too expensive. For cases where the ADMM-DP algorithm converges ($N = 50, 100, 150, 200, 250$ for FF48 and $N = 300, 400$ for S&P 500), the three ADMM algorithms do not dominate each other in final objective values. This makes sense since a global minimum of subproblems may not yield a better final solution for the original problem. Table 4 also shows that the ADMM-Hybrid algorithm did not invoke the DP algorithm in most of the time when solving the $y$ subproblem. In fact, most $y$-subproblems in the ADMM-Hybrid algorithm are solved by the TPPAV algorithm. This indicates that the TPPAV algorithm satisfies the descent properties (9) for most cases. In most cases where the DP algorithm was invoked, the ADMM-Hybrid algorithm required fewer iterations than the ADMM-TPPAV as its subproblem solution is better. We also conclude that solving the subproblems to the global optimum may not yield a better solution for the ADMM algorithm from the comparison of ADMM-DP and ADMM-TPPAV algorithms. Table 4 also shows that the ADMM-TPPAV is the fastest algorithm among the three, thanks to the efficiency.
of the TPPAV algorithm. This, together with its good performance in objective values in Table 3, makes the ADMM-TPPAV a recommended solver for CPT optimization.

6 Empirical Investment Performance

In the empirical analysis, we consider investing in 48 industry indices of the US market (FF48) and investigate the impact of CPT’s four key elements on the optimal portfolio. The historical daily data of 48 industries indices and the risk-free rate, which ranges from 2016-12-12 to 2021-12-11, are collected from Kenneth R. French’s website: http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

Following Tversky and Kahneman (1992), we set the parameters as $\mu = 2.25$, $\alpha = 0.88$, $\gamma = 0.61$, $\delta = 0.69$ and $B = 0$. This setting is named by “CPT $B = 0$” in Table 5, which serves as a benchmark. To investigate the impact of CPT’s key elements on the optimal portfolio, we study four different model settings by varying the corresponding parameter one by one. As shown in Table 5, the second model is named by “CPT $B = r_f$”, which represents that another candidate of the reference point often used in the literature is the risk-free rate. Intuitively, people often feel in loss if his/her investment does not exceed the risk-free rate. By reducing $\alpha$ from 0.88 to 0.5, the third model “$\alpha = 0.5$” investigates how the optimal portfolio will change if the CPT investor becomes much more risk averse. The fourth model “no loss aversion” investigates the effect of loss aversion by setting the loss aversion parameter to be 1. The last model “no probability distortion” investigates the role of probability distortion by setting the probability distortion parameters $\gamma = 1$ and $\delta = 1$.

| Setting name                  | $\mu$ | $\alpha$ | $\gamma$ | $\delta$ | $B$ |
|-------------------------------|-------|-----------|-----------|----------|-----|
| CPT $B = 0$                   | 2.25  | 0.88      | 0.61      | 0.69     | 0   |
| CPT $B = r_f$                 | 2.25  | 0.88      | 0.61      | 0.69     | $r_f$ |
| $\alpha = 0.5$               | 2.25  | 0.5       | 0.61      | 0.69     | 0   |
| no loss aversion              | 1     | 0.88      | 0.61      | 0.69     | 0   |
| no probability distortion     | 2.25  | 0.88      | 1         | 1        | 0   |

At the end of each day from 2017-12-12 to 2021-12-11, we use the running 250 daily returns as the input scenarios, compute the optimal portfolios of the above five models by the ADMM-TPPAV method, and apply the five optimal portfolios for the next day. Then, we obtain 1000 different realizations for each optimal portfolio. The cumulative returns of the five optimal portfolios are reported in Figure 3, where the optimal portfolio of the no loss aversion model (the orange dash line with square) achieves the largest cumulative return. Based on the 1000 different realizations, we can further obtain the mean, volatility, Sharpe ratio, and max drawdown (MD) of the five optimal portfolios, which are summarized in Table 6.
We find that when changing the reference point from 0 to risk-free rate $r_f$, the performances of those two corresponding optimal portfolios are almost the same. This is quite reasonable because $r_f$ is rather small and close to 0. When reducing the risk aversion parameter $\alpha$ from 0.88 to 0.5, the performance of the optimal portfolio remains almost unchanged, which implies that the risk aversion parameter is less important in determining the optimal portfolio under the CPT framework. When reducing the loss aversion from 2.25 to 1, the optimal portfolio achieves the largest cumulative return and mean of returns, but at the cost of highest risk (highest volatility and highest MD), which resulting the smallest Sharpe ratio. Intuitively, loss aversion will make the investor much more risk-averse by introducing a kink around the reference point. Therefore, letting $\mu = 1$, the investor without loss aversion will become less risk averse in terms of taking the riskiest optimal portfolio among the five optimal portfolios. When eliminating the probability distortion, the optimal portfolio achieves the largest Sharpe ratio and the smallest risk (lowest volatility and MD). Such better performance may come from better diversification driven by removing probability distortion, which is confirmed by our diversification analysis below.

Probability distortion will make the investor prefer to gamble and therefore suffer from under-diversification (Barberis, 2013). We investigate the diversification levels of those five different optimal portfolios by looking at the portfolio weights directly. Figure 4 reports the average weights of each portfolio on the top five assets. Once again, there is no big difference between the weights of “CPT $B = 0$” model (Figure 4(a)) and “CPT $B = r_f$” model (Figure 4(b)). The “no probability distortion”
model and “α = 0.5” model give better diversification in terms of the weights of the top five. The “no loss aversion” model gives the most extreme allocation by gambling on one asset heavily.

\[
\text{SSPW} = \sum_{i=1}^{N} \left( x_i - \frac{1}{N} \right)^2 ,
\]

where \( N \) is the number of assets, \( x_i \) is the portfolio weight assigned to asset \( i \) in the portfolio. Clearly, this index measures how the portfolio weight \( x_i \) deviates from the \( 1/N \) portfolio, which is viewed as the best diversified portfolio. Therefore, the lower value of SSPW the higher level of diversification.

On each day from 2017-12-12 to 2021-12-11, we compute SSPW values of the five optimal portfolios and report the corresponding box plots in Figure 5. Compared with the benchmark case “CPT \( B = 0 \)”, removing probability distortion reduces the value of SSPW, therefore increasing the level of diversification. This finding is consistent with the theoretical prediction of Barberis (2013).
Figure 5(c) shows the comparison result for “no loss aversion” and benchmark case. As expected, removing loss aversion makes the investor gamble heavily and therefore produces portfolio concentration, i.e., a large increase in terms of SSPW. Figure 5(b) shows the comparison result by increasing risk aversion parameter from $\alpha = 0.5$ to $\alpha = 0.88$. Increasing the risk aversion parameter and removing loss aversion has a similar effect in determining the value of SSPW. This is quite intuitive. Both increasing the risk aversion parameter and removing loss aversion will make the investor less risk averse, therefore taking a more risky and gambling portfolio.

![Box-plots of five optimal portfolios’ diversification degrees](image)

(a) CPT $B = r_f$  
(b) $\alpha = 0.5$  
(c) no loss aversion  
(d) no probability distortion

Figure 5: The box-plots of five optimal portfolios’ diversification degrees

# Conclusion

In this paper, we proposed the first numerical methodology for CPT portfolio optimization under general assumptions. The optimization problem is difficult to solve because of its S-shaped utility and inverse-S-shaped probability distortion function. To handle this, we introduce an auxiliary variable to represent the historical return of the portfolio and develop an ADMM using the problem structure. However, the difficulty remains in a subproblem of the ADMM that minimizes the negative of the expected utility subject to a chain constraint. We then propose a DP method that returns the global optimal solution of the subproblem. One disadvantage of the DP method is its high computational cost. Based on the celebrated PAV algorithm, this motivates us to design a TPPAV algorithm that returns a local minimizer of the subproblem. In practice, the TPPAV sometimes produces a solution with a larger objective value that does not satisfy a descent property, although it works well in most cases in practice. To remedy this, we propose a hybrid method that combines the DP and TPPAV algorithms. Our numerical experiments show that the ADMM based on the TPPAV algorithm or hybrid algorithm works well in terms of effectiveness and efficiency. We further do an empirical study of CPT portfolio optimization using the proposed method, which has not been done before due to the absence of solvers. Our empirical study reveals the impacts of CPT’s parameters on the optimal portfolios, which is very useful to understand the functions of CPT’s parameters. In future work, we expect to design a method that can find the global optimal solution for CPT optimization.
A Proof of Theorem 1

Proof. In the \((k+1)\)-th iteration, \(\mathbf{x}^{k+1}\) minimizes \(L_\sigma(\mathbf{x}, \mathbf{y}^k; \lambda^k)\). Because \(L_\sigma(\mathbf{x}, \mathbf{y}^k; \lambda^k)\) is a strongly convex function of \(\mathbf{x}\), we have

\[
L_\sigma(\mathbf{x}^k, \mathbf{y}^k; \lambda^k) - L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^k; \lambda^k) \geq \nabla \nabla_x L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^k; \lambda^k)^T (\mathbf{x}^k - \mathbf{x}^{k+1}) + \frac{\sigma \rho}{2} \|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2, \tag{24}
\]

where \(\rho > 0\) is the minimal eigenvalue of \(R^T R\). Because \(\mathbf{x}^{k+1}\) is the global minimum of the \(\mathbf{x}\)-subproblem and \(\mathbf{x}_k \in \mathcal{X}\), we must have

\[
\nabla \nabla_x L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^k; \lambda^k)^T (\mathbf{x}^k - \mathbf{x}^{k+1}) \geq 0. \tag{25}
\]

Combining (24) and (25) gives

\[
L_\sigma(\mathbf{x}^k, \mathbf{y}^k; \lambda^k) - L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^k; \lambda^k) \geq \frac{\sigma \rho}{2} \|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2. \tag{26}
\]

Note that the update (5c) is equivalent to

\[
\mathbf{y}^{k+1} - R\mathbf{x}^{k+1} = -\frac{1}{\sigma} (\lambda^k - \lambda^{k+1}). \tag{27}
\]

Thus we have

\[
L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}; \lambda^k) - L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}; \lambda^{k+1}) = (\lambda^k - \lambda^{k+1}, \mathbf{y}^{k+1} - R\mathbf{x}^{k+1}) = -\frac{1}{\sigma} \|\lambda^k - \lambda^{k+1}\|^2. \tag{28}
\]

Summing up (26), (9) and (28) gives rise to

\[
L_\sigma(\mathbf{x}^k, \mathbf{y}^k; \lambda^k) - L_\sigma(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}; \lambda^{k+1}) \geq \frac{\sigma \rho}{2} \|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2 - \frac{1}{\sigma} \|\lambda^k - \lambda^{k+1}\|^2. \tag{29}
\]

Under the assumption that the sequence \(\{\lambda^k\}\) is bounded and the set \(\mathcal{X}\) is bounded, it follows from (27) that \(\{\mathbf{y}^k\}\) is also bounded. Therefore \(L_\sigma(\mathbf{x}^k, \mathbf{y}^k; \lambda^k)\) is also bounded. Then summing up both sides of (29) for \(k\) from 1 to infinity, we have

\[
\frac{\sigma \rho}{2} \sum_{k=1}^{\infty} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 < \infty + \frac{1}{\sigma} \sum_{k=1}^{\infty} \|\lambda^{k+1} - \lambda^k\|^2
\]

This, together with the assumption that \(\sum_{k=1}^{\infty} \|\lambda^{k+1} - \lambda^k\|^2 < \infty\), implies \(\sum_{k=1}^{\infty} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 < \infty\), which further yields \(\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \to 0\) for \(k \to \infty\). Using \(\sum_{k=1}^{\infty} \|\lambda^{k+1} - \lambda^k\|^2 < \infty\) from Assumption 1, we have \(\lambda^{k+1} - \lambda^k \to 0\). Then (27) further implies

\[
\mathbf{y}^k - R\mathbf{x}^k \to 0. \tag{30}
\]

Due to \(\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \to 0\) and \(\lambda^{k+1} - \lambda^k \to 0\), (27) implies that

\[
\|\mathbf{y}^{k+1} - \mathbf{y}^k\| \leq \|R(\mathbf{x}^{k+1} - \mathbf{x}^k)\| + \frac{1}{\sigma} \|\lambda^k - \lambda^{k+1}\| - (\lambda^{k-1} - \lambda^k)\| \to 0. \tag{31}
\]
As the $x$-subproblem is globally solved, we must have
\[
0 \in -\sigma R^T \left( y^k - R x^{k+1} + \frac{x^k}{\sigma} \right) + N_\chi(x^{k+1}).
\] (32)

Note also that (10) is equivalent to
\[
0 \in R^T \partial \Omega(y^{k+1}) + \sigma R^T \left( y^k - R x^{k+1} + \frac{x^k}{\sigma} \right) + \sigma R^T (y^{k+1} - y^k).
\] (33)

From (32) and (33), we deduce that
\[
0 \in R^T \partial \Omega(y^{k+1}) + N_\chi(x^{k+1}) + \sigma R^T \left( y^{k+1} - y^k \right).
\]
This, together with (31), Lemma 2 and the fact that the normal cone of a closed set is outer semicon-
tinuous set-valued mapping (Rockafellar and Wets, 2009, Proposition 6.6), yields $0 \in R^T \partial \Omega(x^*) + N_\chi(x^*)$, where $(x^*, y^*)$ is any accumulation point of $\{(x^k, y^k)\}$. Note also that (30) implies $y^* = Rx^*$. The proof is completed.

\[\square\]

References

Ahuja, R. K. and Orlin, J. B. (2001). A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. *Operations Research*, 49(5):784–789.

Allais, M. (1953). Le comportement de l’homme rationnel devant le risque: critique des postulats et axiomes de l’école américaine. *Econometrica*, pages 503–546.

Ayer, M., Brunk, H. D., Ewing, G. M., Reid, W. T., and Silverman, E. (1955). An empirical distribution function for sampling with incomplete information. *The annals of mathematical statistics*, pages 641–647.

Bai, X., Sun, J., and Zheng, X. (2021). An augmented lagrangian decomposition method for chance-constrained optimization problems. *INFORMS Journal on Computing*, 33(3):1056–1069.

Barberis, N. (2013). The psychology of tail events: Progress and challenges. *American Economic Review*, 103(3):611–16.

Barberis, N. and Huang, M. (2008). Stocks as lotteries: The implications of probability weighting for security prices. *American Economic Review*, 98(5):2066–2100.

Barberis, N. and Xiong, W. (2009). What drives the disposition effect? An analysis of a long-standing preference-based explanation. *Journal of Finance*, 64(2):751–784.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (2006). *Nonlinear programming: theory and algorithms*. John Wiley & Sons.

Beck, A. (2017). *First-order methods in optimization*. SIAM.

Bernard, C. and Ghossoub, M. (2010). Static portfolio choice under cumulative prospect theory. *Mathematics and financial economics*, 2(4):277–306.

Best, M. J. and Chakravarti, N. (1990). Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1):425–439.
Best, M. J., Chakravarti, N., and Ubhaya, V. A. (2000). Minimizing separable convex functions subject to simple chain constraints. *SIAM Journal on Optimization*, 10(3):658–672.

Bi, X., Yuan, L., Cui, Z., Fan, J., and Zhang, S. (2021). Optimal investment problem under behavioral setting: A lagrange duality perspective. *Available at SSRN 3801926*.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.

Brunk, H., Barlow, R. E., Bartholomew, D. J., and Brenner, J. M. (1972). Statistical inference under order restrictions.(the theory and application of isotonic regression). Technical report, Missouri Univ Columbia Dept of Statistics.

Chen, Y., Li, C., and Lu, Z. (2022). An admm based parallel approach for fund of fund construction. *Algorithms*, 15(2):35.

Clarke, F. H. (1990). *Optimization and nonsmooth analysis*. SIAM.

Costa, G. and Kwon, R. H. (2020). Generalized risk parity portfolio optimization: An admm approach. *Journal of Global Optimization*, 78(1):207–238.

Cui, X., Sun, X., Zhu, S., Jiang, R., and Li, D. (2018). Portfolio optimization with nonparametric value at risk: A block coordinate descent method. *INFORMS Journal on Computing*, 30(3):454–471.

Cui, Z., Lee, C., Zhu, L., and Zhu, Y. (2021). Non-convex isotonic regression via the myersonian approach. *Statistics & Probability Letters*, 179:109210.

De Giorgi, E. and Legg, S. (2012). Dynamic portfolio choice and asset pricing with narrow framing and probability weighting. *Journal of Economic Dynamics and Control*, 36(7):951–972.

Glowinski, R. and Marrocco, A. (1977). Numerical solution of two-dimensional magnetostatic problems by augmented lagrangian methods. *Computer methods in applied mechanics and engineering*, 12(1):33–46.

Goetzmann, W. N. and Kumar, A. (2008). Equity portfolio diversification. *Review of Finance*, 12(3):433–463.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

He, X. D. and Zhou, X. Y. (2011a). Portfolio choice under cumulative prospect theory: An analytical treatment. *Management Science*, 57(2):315–331.

He, X. D. and Zhou, X. Y. (2011b). Portfolio choice via quantiles. *Mathematical Finance*, 21(2):203–231.

Hens, T. and Mayer, J. (2014). Cumulative prospect theory and mean variance analysis: A rigorous comparison. *Swiss finance institute research paper*, (14-23).

Jin, H. and Zhou, X. Y. (2008). Behavioral portfolio selection in continuous time. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 18(3):385–426.

Jorion, P. (1997). *Value at risk: the new benchmark for controlling market risk*. Irwin Professional Pub.

Kahneman, D. and Tversky, A. (1979). Prospect theory: An analysis of decisions under risk. *Econometrika*, 47:263–291.

Konno, H. and Koshizuka, T. (2005). Mean-absolute deviation model. *Iie Transactions*, 37(10):893–900.
Konno, H. and Suzuki, K.-i. (1995). A mean-variance-skewness portfolio optimization model. *Journal of the Operations Research Society of Japan*, 38(2):173–187.

Lai, Z.-R., Yang, P.-Y., Fang, L., and Wu, X. (2018). Short-term sparse portfolio optimization based on alternating direction method of multipliers. *The Journal of Machine Learning Research*, 19(1):2547–2574.

Levy, H. and Levy, M. (2004). Prospect theory and mean-variance analysis. *Review of Financial Studies*, 17(4):1015–1041.

Luxenberg, E., Schiele, P., and Boyd, S. (2022). Portfolio optimization with cumulative prospect theory utility via convex optimization. *arXiv preprint arXiv:2209.03461*.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

Pirvu, T. and Schulze, K. (2012). Multi-stock portfolio optimization under prospect theory. *Mathematics and Financial Economics*, pages 1–26.

Rockafellar, R. T., Uryasev, S., et al. (2000). Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42.

Rockafellar, R. T. and Wets, R. J.-B. (2009). *Variational analysis*, volume 317. Springer Science & Business Media.

Rüschendorf, L. and Vanduffel, S. (2020). On the construction of optimal payoffs. *Decisions in Economics and Finance*, 43(1):129–153.

Shen, Y., Wen, Z., and Zhang, Y. (2014). Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29(2):239–263.

Shi, Y., Cui, X., and Li, D. (2015). Discrete-time behavioral portfolio selection under cumulative prospect theory. *Journal of Economic Dynamics and Control*, 61:283–302.

Srivastava, S., Aggarwal, A., and Mehra, A. (2022). Portfolio selection by cumulative prospect theory and its comparison with mean-variance model. *Granular Computing*, pages 1–14.

Strömberg, U. (1991). An algorithm for isotonic regression with arbitrary convex distance function. *Computational Statistics & Data Analysis*, 11(2):205–219.

Tversky, A. and Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5(4):297–323.

van Bilsen, S. and Laeven, R. J. (2020). Dynamic consumption and portfolio choice under prospect theory. *Insurance: Mathematics and Economics*, 91:224–237.

Von Neumann, J. and Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton university press.

Wei, P. (2018). Risk management with weighted var. *Mathematical Finance*, 28(4):1020–1060.

Xu, Y., Yin, W., Wen, Z., and Zhang, Y. (2012). An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, 7(2):365–384.

Yu, Z., Chen, X., and Li, X. (2020). A dynamic programming approach for generalized nearly isotonic optimization. *arXiv preprint arXiv:2011.03305*.

Zou, B. and Zagst, R. (2017). Optimal investment with transaction costs under cumulative prospect theory in discrete time. *Mathematics and Financial Economics*, 11(4):393–421.