Portfolio Optimization with Trading Costs

Mingyu Liu

November 20, 2020

Abstract

In this report, we have compared the efficiencies of eight modelling package + solver combinations in solving Markowitz optimization models with none, linear, quadratic and regressed real-world trading cost models. `(Pyomo, IPOPT)` is our recommended open-sourced combination for its high efficiency, low variance in solving time and the ability to handle large-scale problems. The report is structured as follows: In Section 1, we begin with a description of the setup of the evaluation experiments. Then, in Section 2, we present and interpret the results from the experiments. Next, we provide extra details about the recommended `IPOPT` solver in Section 3. In Section 4, we discuss qualitatively some additional properties of the modelling packages, before concluding in Section 5 with discussions about the limitations of this investigation and directions for future work.

1 Problem Definition

1.1 Notations

We consider a universe of n assets. Let $w = (w_1, \ldots, w_n)$ be the weight vector of assets in the portfolio. The expected return of portfolio w is given by

$$\mu(w) = \mathbb{E}\left[\sum_{i=1}^{n} w_i R_i\right] = w^\top \mu$$

where $R = (R_1, \ldots, R_n)$ is the vector of asset return random variables and $\mu = (\mu_1, \ldots, \mu_n)$ is the vector of expected asset returns. The mean-variance optimization framework of Markowitz [18] consists of maximizing the expected return $\mu(w)$ for a given level of the volatility $\sigma(w)$

$$\sigma^2(w) = \mathbb{E}\left[\left(\sum_{i=1}^{n} w_i R_i - \mu(w)\right)^2\right] = w^\top \Sigma w$$

where Σ is the covariance matrix of asset returns. The mean-variance optimization framework can then be formulated as standard quadratic programming (QP) problem

$$\begin{aligned} \text{minimize} \quad & -\left(w^\top \mu - \frac{1}{2} w^\top \Sigma w\right) \\ \text{s.t.} \quad & \begin{cases} \mathbf{1}_n^\top w = 1 \\ \mathbf{0}_n < w < \mathbf{1}_n \end{cases} \end{aligned}$$

The budget constraint implies that the wealth is entirely invested and the second constraint indicates that we adopt a long-only strategy.

1.2 Problem Formulation

Let us now introduce the trading cost. We use $\tilde{w}$ to denote the current portfolio and $\mathcal{C}(w \mid \tilde{w})$ the cost of rebalancing the current portfolio $\tilde{w}$ towards the portfolio w. Assuming that the trading costs are deterministic, the net return is equal to the gross return minus the trading costs

$$\mu(w \mid \tilde{w}) = \mu(w) - \mathcal{C}(w \mid \tilde{w}) = w^\top \mu - \mathcal{C}(w \mid \tilde{w})$$

while the variance is assumed to be unaffected.[1] The optimization problem can be formulated most generally as

$$\begin{aligned} \text{minimize} \quad & -\left[\left(w^\top \mu - \mathcal{C}(w \mid \tilde{w})\right) - \frac{1}{2} w^\top \Sigma w\right] \\ \text{s.t.} \quad & \begin{cases} \mathbf{1}_n^\top w + \mathcal{C}(w \mid \tilde{w}) = 1 \\ \tilde{w} + \Delta w^+ - \Delta w^- = w \\ \mathbf{0}_n \leq w \leq \mathbf{1}_n \end{cases} \end{aligned}$$

The first constraint is known as the budget constraint. The second constraint is known as the turnover constraint, where we use $\Delta w^- = \max(\tilde{w} - w, 0)$ and $\Delta w^+ = \max(w - \tilde{w}, 0)$ to represent the sale and purchase of assets. By construction, the vectors Δw^+ and Δw^- are orthogonal ${\Delta w^-}^\top \Delta w^+ = 0$. The trading cost function is always positive as trades move price against buyers and sellers. The third constraint indicates that our strategy is long-only. It should be pointed out that w represents the effective portfolio holding after trading costs are subtracted. Their sum does not equal to one unless the expected trading cost is zero. The proposed formulation directly follows Chen, Pierre, et al [7]. However, this formulation suffers an obvious weakness – the trading cost term $\mathcal{C}(w \mid \tilde{w})$ and the overall objective function is not necessarily convex. We rewrite the objective function by

$$\begin{aligned} \underset{w \in \mathcal{R}^N}{\text{minimize}} \quad & -\left[\left(w^\top \mu - \mathcal{C}(w \mid \tilde{w})\right) - \frac{1}{2} w^\top \Sigma w\right] \\ \iff & -\left[1 + \left(w^\top \mu - \mathcal{C}(w \mid \tilde{w})\right) - \frac{1}{2} w^\top \Sigma w\right] \quad \text{(initial wealth + return - costs)} \\ \iff & -\left[w^\top (\mathbf{1}_n + \mu) - \frac{1}{2} w^\top \Sigma w\right] \end{aligned}$$

The return term in the objective is affine. The quadratic term can be made convex by restricting the covariance matrix to be positive definite (which we will discuss in detail later). Overall, the objective function is convex and can be tackled by a variety of non-linear optimization solvers [5, 15, 17]. However, it is evident that the the vector w of portfolio weights does not add up to one and hence does not represent the actual tradable allocation. An alternative formulation with respect to the tradable weights $\hat{w}$ and its relationship to the current formulation is discussed in Appendix B.

1.3 Trading Costs

Linear Cost

First, we consider a simple proportional trading cost model where the unit cost of trading does not vary with the size of the portfolio traded. We distinguish between the bid and ask price and use

[1] In general, the trading costs $\mathcal{C}(w \mid \tilde{w})$ should be treated as random variables and impact both the expected return

b_i^- and b_i^+ to denote the bid and ask trading cost for each unit of asset i traded. The trading cost for any asset satisfies

$$c_i\left(w \mid \tilde{w}\right) = \begin{cases} b_i^- \cdot (\tilde{w}_i - w_i) & \text{if } w_i < \tilde{w}_i \\ 0 & \text{if } w_i = \tilde{w}_i \\ b_i^+ \cdot (w_i - \tilde{w}_i) & \text{if } w_i > \tilde{w}_i \end{cases}$$

or in shorthand notation $\mathcal{C}\left(w \mid \tilde{w}\right) = \Delta w^{-\top} b^- + \Delta w^{+\top} b^+$. For simplicity, we have use the same unit cost $b_i^- = 1\%$ and $b_i^+ = 2\%$ across all assets. Since the trading cost $\mathcal{C}\left(w \mid \tilde{w}\right)$ is a nonlinear function of w, the problem is not readily a standard QP problem. However, it can be reformulated as one with respect to the augmented variables $x = (w, \Delta w^-, \Delta w^+)$ [7].

Quadratic Cost

The linear trading cost model is generally applicable to small investors. However, for larger institutional traders, models in which the cost of trading varies with the size of the traded portfolio are more realistic, because their trades can move price. In this section, we consider the model where the price impact function is proportional to the trade size. In this situation, we deduce that

$$c_i\left(w \mid \tilde{w}\right) = \begin{cases} b_i^- \cdot (\tilde{w}_i - w_i) + k_i^- \cdot (\tilde{w}_i - w_i)^2 & \text{if } w_i < \tilde{w}_i \\ 0 & \text{if } w_i = \tilde{w}_i \\ b_i^+ \cdot (w_i - \tilde{w}_i) + k_i^+ \cdot (w_i - \tilde{w}_i)^2 & \text{if } w_i > \tilde{w}_i \end{cases}$$

In shorthand, $\mathcal{C}\left(w \mid \tilde{w}\right) = \Delta w^{-\top} b^- + \Delta w^{-\top} \mathrm{K}^- \Delta w^- + \Delta w^{+\top} b^+ + \Delta w^{+\top} \mathrm{K}^+ \Delta w^+$ where $\mathrm{K}^- = \operatorname{diag}\left(k_1^-, \ldots, k_n^-\right)$ and $\mathrm{K}^- = \operatorname{diag}\left(k_1^+, \ldots, k_n^+\right)$. By construction, the quadratic costs have a more adverse effect on the portfolio's return, especially if the rebalancing is significant.

By rearranging the problem again with respect to the augmented variable $x = (w, \Delta w^-, \Delta w^+)$ according to Chen, Pierre, et al., the optimization framework can be reformulated into a standard quadratically constrained quadratic program (QCQP) problem [7]. It should be pointed out that the quadratic constraint in the problem is an equality constraint, meaning the problem is in general non-convex. However, as the covariance matrix is positive positive semi-definite, it can be solved by the SDP or SOCP relaxations [13].

Generic Cost

In this section, we introduce the trading cost and price impact model developed by Frazzini, Andrea, et al. based on 1.7 trillion dollars of real-world live trade execution data from a large institutional money manager across 21 developed equity markets over a 19-year period [11]. The model takes into consideration the trade type, market characteristics, stock characteristics, trade size and time.

---

and the volatility of the portfolio. The mean and standard deviation of the trading costs are

$$\begin{aligned} \mu\left(w \mid \tilde{w}\right) &= \mathbb{E}\left[R\left(w \mid \tilde{w}\right) - \mathcal{C}\left(w \mid \tilde{w}\right)\right] \\ &= \mu\left(w\right) - \mu_{\mathcal{C}}\left(w \mid \tilde{w}\right) \end{aligned}$$

$$\begin{aligned} \sigma^2(w \mid \tilde{w}) &= \mathbb{E}\left[\left(R(w \mid \tilde{w}) - \mu(w \mid \tilde{w})\right)^2\right] \\ &= \mathbb{E}\left[\left(R(w) - \mu(w) + \mu_{\mathcal{C}}(w \mid \tilde{w}) - \mathcal{C}(w \mid \tilde{w})\right)^2\right] \\ &= \sigma^2(w) + \sigma_{\mathcal{C}}^2(w \mid \tilde{w}) - 2\rho_{\mathcal{C}}(w \mid \tilde{w}) \cdot \sigma(w)\sigma_{\mathcal{C}}(w \mid \tilde{w}) \end{aligned}$$

where $\mu_{\mathcal{C}}(w \mid \tilde{w}) = \mathbb{E}[\mathcal{C}(w \mid \tilde{w})]$ is the expected cost of rebalancing, $\sigma_{\mathcal{C}}(w \mid \tilde{w})$ is the standard deviation of $\mathcal{C}\left(w \mid \tilde{w}\right)$ and $\rho_C(w \mid \tilde{w})$ is the correlation between the gross return $R(w)$ and the trading costs $\mathcal{C}\left(w \mid \tilde{w}\right)$. However, the budget constraints would become stochastic $\mathbf{1}_n^\top w + \mathcal{C}(w \mid \tilde{w}) = 1$ making the problem intractable.

The expected unit market impact in basis point is given by the following equation, using the same notations as in the original paper

$$\bar{c}_i\left(w \mid \tilde{w}\right) = p_0 + p_1 t + p_2 \log\left(\mathrm{ME}_{t-1}\right) + p_3 x_{i,t-1} + p_4 \operatorname{sign}\left(x_{i,t-1}\right)\sqrt{|x_{i,t-1}|} + \theta_5 \sigma_{t-1}^{IV} + \theta_6 \mathrm{VIX}_{t-1}$$

where $x = 100 \times m/dtv$ is the signed dollar volume (m) as a fraction (in %) of the stock's average past one-year dollar volume dtv. The contemporaneous market and the matched-characteristic benchmark (DGTW) return are dropped as they are unobservable and have zero expectation at a daily frequency. The remaining variables are all *ex-ante* or choice variables that can be used to estimate the expected trading cost of a portfolio out of sample. The p_1 term is a linear time trend that captures the changes in aggregate trading costs over time. ME in the p_2 term stands for the market value of equity in million USD. The p_3 and p_4 term represents the linear and square root relationship between price impact and size of trade. The p_5 term is the idiosyncratic volatility of the firm's equity return. Finally, the p_6 term captures the market volatility. Details aside, the unit trading cost can be minimally parameterized as

$$\bar{c}_i\left(w \mid \tilde{w}\right) = a' + b' x_{i,t-1} + c' \operatorname{sign}\left(x_{i,t-1}\right)\sqrt{|x_{i,t-1}|}$$

In this equation, $x_{i,t-1}$ can be related back to Δw_i via exposure and the firm's market price. The trading cost of trade can then be written as

$$c_i\left(w \mid \tilde{w}\right) = a \cdot |\Delta w_i| + b \cdot \Delta w_i^2 + c \cdot |\Delta w_i|^{3/2}$$

This cost function is convex when $a, b, c >= 0$. However, several modelling tools do not support power operators with arbitrary power. In these cases, we change the last term to a quadratic term and subsume it to the second term.

2 Experimental Setups

In this investigation, we have developed two evaluation schemes to compare the efficiencies of eight modelling package + solver combinations suited for solving Markowitz' mean-variance portfolio optimization problems with none, linear, quadratic and generic trading cost models [10].[2] The combinations included in the evaluation are `(Scipy, SLSQP)`, `(CVXPY, ECOS)`, `(CVXPY, GUROBI)`, `(CVXPY, MOSEK)`, `(Pyomo, IPOPT)`, `(Pyomo, SCIP)`, `(Pyomo, BONMIN)`, `(GUROBI, GUROBI)`. They are selected from a pool of twenty that satisfies the following two requirements: i) able to solve optimization models with non-linear (quadratic and generic) costs up to 500 assets ii) the solving times for 500 assets are below 30 seconds. The two evaluation schemes are described as follows:

- **Test (a):** The optimal weights are solved for on a fixed trading day with different numbers of assets ranging from 2-500. The initial portfolio weights are set to zero, assuming no exposure. The test is repeated 12 times to get estimations of the variances in solving times (assuming normal distributions).
- **Test (b):** The optimal weights are solved for on every trading day consecutively for a two-year period from Jan-2015 to Dec-2016, with a fixed pool of 500 assets. The initial portfolio weights on a particular trading day are equal to the allocated weights from the previous trading day. An equal-weighted portfolio is assumed at the beginning of the test (2015-01-02).

[2]The generic cost model is regressed using 1.7 trillion dollars of live trade execution data from a large institutional money manager across 21 developed equity markets over 19 years period. It is designed to measure the real-world trading costs and price impact function of a large trader. The model takes into account factors including trade type, stock characteristics, trade size and time.

In both tests, we warm-start the solvers with the initial portfolio weights described. The relative error tolerances across all solvers are standardized to `1e-6`.

2.1 Expected Return & Covariance

In this subsection, we provide further details on how the vectors of expected returns and the covariance matrices in our portfolio models are calculated. We use the CAPM model to estimate the expected return r_i of an asset i

$$r_i = r_f + \beta_i \left(r_m - r_f\right)$$

where r_f represents the risk-free rate, and r_m represents the market return which we obtain from a simulated market index. The covariance matrix is constructed using the formula [3]

$$\mathrm{cov}_{ij} = \beta_i \beta_j \cdot \mathrm{var}\left(r_m - r_f\right) + \delta_{ij} \cdot \mathrm{ivol}_i^2$$

where δ_{ij} is the Kronecker delta and ivol_i is the idiosyncratic risk of asset i measured by the variance of the CAPM model residues. This form ensures that the covariance matrix is positive-definite, hence the corresponding mean-variance objective function is convex. On the other hand, a quadratic form with semi-definite or indefinite covariance matrix would produce non-convex objectives, which many solvers cannot solve or solve much more slowly.

Simulated Market Index

We use a simulated random walk time series with returns correlated to the S&P500 returns as the market index used in the CAPM and covariance formulae. We treat the simulated return and the S&P500 return on a given trading day as normally distributed random variables X_1, X_2 with means μ_1, μ_2 and variances σ_{11}, σ_{22}. σ_{12}^2 denotes the covariance between the two random variables (so does σ_{21}^2).

$$Z = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \qquad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$$

The conditional distribution of X_1 given knowledge of the observed S&P500 index x_2 is a normal distribution with [4]

$$\text{Mean} = \mu_1 + \frac{\sigma_{12}}{\sigma_{22}}\left(x_2 - \mu_2\right) \qquad\qquad \text{Variance} = \sigma_{11} - \frac{\sigma_{12}^2}{\sigma_{22}}$$

We let the means and variances of the two return variables to be identical (i.e. $\mu_1 = \mu_2 = \mu$ and $\sigma_{11} = \sigma_{22} = \sigma$), and the covariance $\sigma_{12}^2 = \sigma_{21}^2 = \rho \cdot \sigma^2$ with a correlation coefficient $\rho = 0.2$. The conditional distribution of X_1 given x_2 then becomes

$$\text{Mean} = \mu + \sqrt{\rho}\left(x_2 - \mu\right) \qquad\qquad \text{Variance} = (1 - \rho) \cdot \sigma$$

where μ and σ are estimated using the sample mean and sample standard deviation of the real S&P500 returns with a 63-day moving average centered at the day of interest.

In the CAPM and covariance formulae, we use the market return on the same day when the portfolio is traded and the daily return is calculated. It is obvious from this construction that our trading

[3] the variance $\mathrm{var}\left(r_f\right)$ of the risk-free rate could also be included, although its influence will ve tiny.

[4] [Reference] In general, we would have a $p \times 1$ random vector $\mathbf{Z}$ which we can partition it into two random vectors $\mathbf{X}_1$ and $\mathbf{X}_2$ where $\mathbf{X}_1$ is a $p_1 \times 1$ vector and $\mathbf{X}_2$ is a $p_2 \times 1$ vector. Further, suppose that we partition the mean vector

strategy is *ex-post* – it uses market information after the actual trading occurs and is expected to give high returns and Sharpe ratios unrealistic in real portfolio investment (as we shall see later). However, this approach is entirely valid for the purpose of testing optimization solvers. It saves us the time of developing a good asset pricing model and allows us to focus our attention on evaluating the software.

Note: The data in this investigation are obtained from the WRDS Beta Suite and the WRDS CRSP database. In total, we obtained the returns and the generic cost parameters (e.g. market-capitalization, trading volume etc.) of a pool of 2507 companies that have no missing data over the whole two-year testing period.

2.2 Environmental Configuration

The tests are performed a local Macbook Pro machine equiped with 2.8GHz Intel Quad-Core i7 Processor and 16GB RAM. In running the tests, we have not allowed any parallelization – a single process can sometimes utilze multiple cores and take up more than 50% of the CPU resources. Hence, any parallelization may result in inefficiency in each individual process and skew the evaluation. The whole experiment took around 35 hours to complete.

3 Experimental Results

3.1 Test (a): Efficiencies

In Figure 1, we present a comparison of the efficiencies of the modelling package + solver combinations vs. the number of assets for models with different cost types from test (a). It plots the mean solving times (in log-scale) vs. the number of assets, with the shaded area representing sample standard deviations. The legends are sorted in ascending order of the solving times of the largest optimization problems (i.e. portfolios with 500 assets).

It is discovered that the `(CVXPY, MOSEK)` pair consistently delivers competitive performance across all trading cost types. The `(CVXPY, GUROBI)` is fast for none and linear costs, but its performance deteriorates as the constraints start to involve non-linear terms (quadratic and generic costs).[5] Among the open-sourced solvers, `(Pyomo, IPOPT)` and `(Pyomo, BONMIN)` are the fastest for solving problems with quadratic and generic costs. The solving times scale smoothly with the problem sizes and have low variances. However, the gaps in performance compared to the commercial solvers are evident in the case of none and linear costs. It is also observed that the computational complexity of `(Scipy, SLSQP)` grows faster than the others for all cost types, suggesting that it is unsuitable for large-scale problems.

---

and covariance matrix in a corresponding manner. That is,

$$\mathbf{Z} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix} \qquad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}$$

Any distribution for a subset of variables from a multivariate normal, conditional on known values for another subset of variables, is a multivariate normal distribution. The conditional distribution of $\mathbf{X}_1$ given known values for $\mathbf{X}_2 = \mathbf{x}_2$ is a multivariate normal with:

$$\begin{aligned} \text{mean vector} &= \mu_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{\mathbf{22}}^{-\mathbf{1}} (\mathbf{x_2} - \mu_2) \\ \text{covariance matrix} &= \boldsymbol{\Sigma}_{\mathbf{11}} - \boldsymbol{\Sigma}_{\mathbf{12}}\boldsymbol{\Sigma}_{\mathbf{22}}^{-\mathbf{1}}\boldsymbol{\Sigma}_{\mathbf{21}} \end{aligned}$$

[5] Xpress is another well-regarded commercial solver developed by FICO®. However, our problem sizes are larger than the limit set by the community license available to students. It was therefore dropped from our comparison.

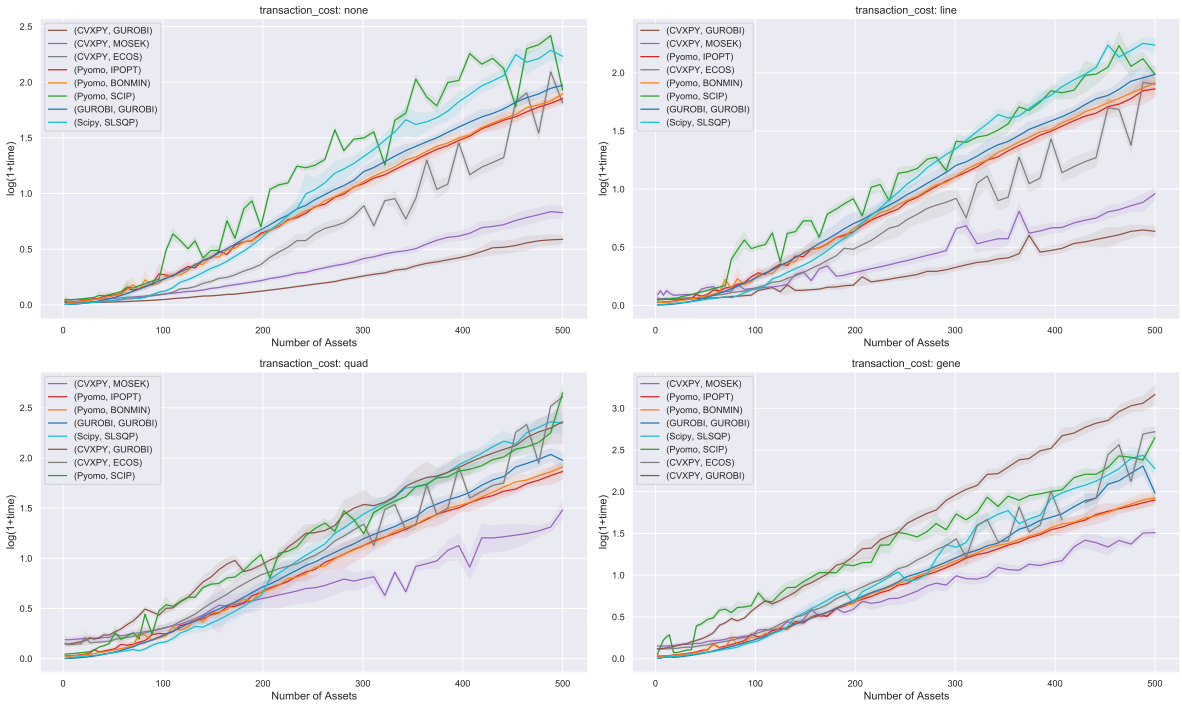

Figure 1: The mean solving times (in log-scale) vs. the number of assets (with sample standard deviations represented by the shaded area). Legends are sorted in ascending order of the solving times. MOSEK (commercial) is the solver that consistently delivers superior performance for all types of trading costs. Among the open-sourced solvers, `(Pyomo, IPOTP)` and `(Pyomo, BONMIN)` have very good performance for complex non-linear constraints (i.e. quadratic and generic cost), and outperforms some commercial solvers including `GUROBI`.

It should be noted that a commercial solver usually comes with a set of algorithms and can automatically switch between them depending on the problem types (e.g. `Xpress` is built with the primal simplex, dual simplex and Newton barrier algorithm for each of the LP, QP and QCQP/SOCP problems) [9, 12, 19]. On the other hand, an open-sourced solver usually bases upon a single algorithm (e.g. `IPOPT` uses the interior point algorithm alone). Therefore, if we decide to use an open-sourced solver for an optimization problem, it is crucial to identify its problem type before selecting the appropriate solver. For instance, if we were are interested in solving a portfolio model with none or linear trading costs, we might instead recommend a dedicated QP solver such as `OSQP` or `qpOASES` [1, 8].

3.2 Test (b): Efficiencies

Figure 2 contains the boxplots of solving times for the four types of trading costs from test (b). Legends are again sorted in ascending order of the solving times. The rankings of the solver efficiencies are quite similar to that from test (a). `(Pyomo, IPOPT)` and `(Pyomo, BONMIN)` are still the most efficient open-sourced combinations for quadratic and generic costs. We point out that the `BONMIN` solver is built based upon `IPOPT` (and another solver `Cbc`). `BONMIN` has the additional benefit of supporting mixed integer programming (MIP) – a functionality that we do not need. Therefore,

However, for problem sizes solvable by the community license, it achieves pretty competitive results comparable to MOSEK.

we could simply use `IPOPT`, without potentially experiencing from the overheads introduced by `BONMIN`.

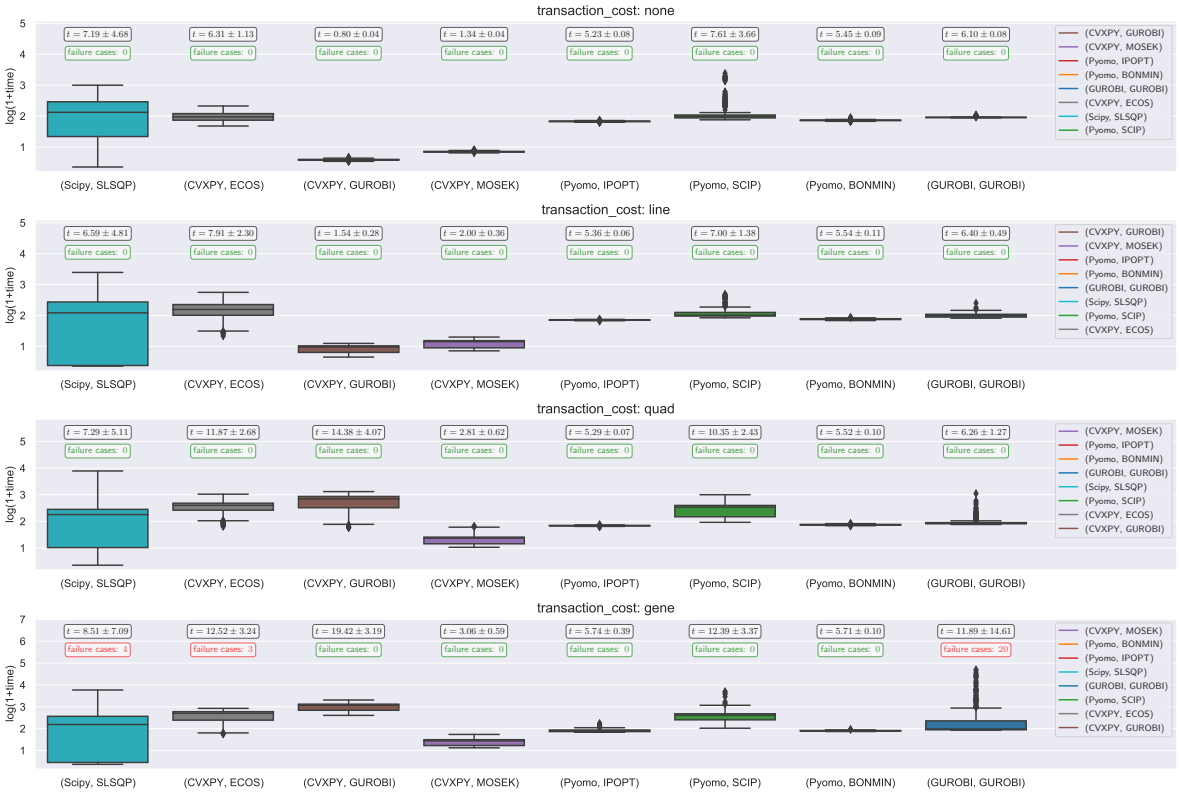

Figure 2: Boxplots of the solving times (in log-scale) of the modelling package + solver pairs. Legends are sorted in ascending order of solving times.

Another interesting observation is that the solving times of `(Scipy, SLSQP)` have relatively large variances that are not observed in Figure 1 where the same optimizations problem are solved repeatedly, suggesting that the solving times are likely to be related to the problem inputs. It is discovered that the solving times of `(Scipy, SLSQP)` have a high correlation (none: 0.762, line: 0.800, quad: 0.736, gene: 0.629) to the changes in weights $\sum_i |w_{i,t+1} - w_{i,t}\,(1 + r_{i,t+1})|$ before and after rebalancing. High correlation is also observed for problems modelled with `CVXPY`. We can take advantage of this property to offer the solvers good warm starts when the change in weights are expected to be small. However, the correlations are not for `Pyomo` and `GUROBI`.

3.3 Test (b): Backtesting

Figure 3 plots the cumulative returns and daily returns from our trading strategy over the two-year backtesting period. It also includes a summary of the i) average daily return ii) annualized Sharpe ratio iii) average daily turnover over the testing period. The daily returns and Sharpe ratio are unrealistically high. This is not surprising given the extensive use of future information in our strategy. The trading costs have been constructed such that `gene > quad > line > none` given the same rebalancing, which explains the order of the cumulative returns.

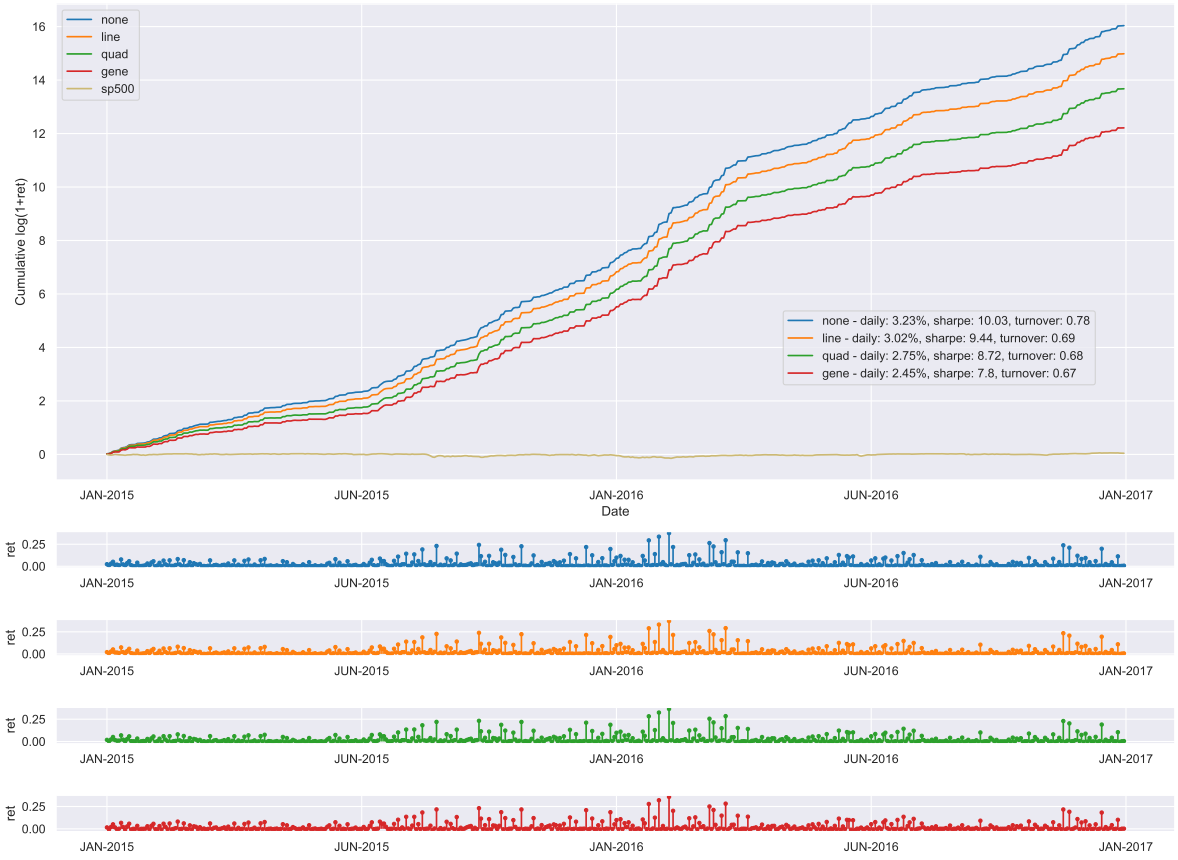

Figure 3: Cumulative returns and daily returns with the four types of trading costs, including a summary of the i) average daily return ii) annualized Sharpe ratio iii) average daily turnover during the two-year testing period

3.4 Precision

The relative error tolerances have been standardized to `1e-6` across all solvers. The optimal solutions from test (a) and cummulative returns from test (b) obtained from different solvers demonstrate high consistency. Therefore, no particular solver is favored from a precision point of view. (Figure 4 – 5 in Appendix B).

4 Additional Details about IPOPT

Although this investigation mainly concerns with problems with convex quadratic objectives, the capability of Pyomo extends much beyond that. From the documentation, IPOPT (Interior Point OPTimizer) is a software package for large-scale non-linear optimization designed to find (local) solutions of mathematical optimization problems of the form

$$\underset{x \in R^n}{\text{minimize}} \quad f(x) \qquad \text{s.t.} \quad \begin{cases} g_L \leq g(x) \leq g_U \\ x_L \leq x \leq x_U \end{cases}$$

where $f(x) : R^n \to R$ is the objective function and $g(x) : R^n \to R^m$ are the constraint functions. The vectors g_L and g_U denote the lower and upper bounds on the constraints, and the vectors x_L

and x_U are the bounds on the variables x. Equality constraints can be formulated by setting the components of the lower and upper bounds to the same value. The functions $f(x)$ and $g(x)$ can be non-linear and non-convex, but should be twice continuously differentiable [22]. The mathematical details of the algorithm can be found in several publications [2, 3, 14].

5 Additional Details about the Modelling Packages

In this section, we provide a qualitative evaluation of the modelling packages in terms of the flexibility, the solver support and the ease of use.

5.1 Flexibility

Among all modelling packages, `Scipy` allows the most generic problem formulation – there is almost no restriction on the expressions of the objectives and constraints, any function in `numpy` can be used. Whether optimal solutions can be found or not is left to the solvers to decide. `Pyomo` and `GUROBI` are also very flexible, but do not allow non-continuously differentiable functions such as `min`, `max` and `abs` in the objectives. However, these operations can be included in constraints, meaning that if we want to optimize objectives with non-continuously differentiable terms, we usually need to introduce auxiliary variables and relate them to the main variables via constraints.

`CVXPY` imposes the most strict restrictions on the problem. Objective functions and constraints needs have well-defined curvatures according to a set of composition rules. A standard problem solvable in `CVXPY` need to follow the DCP rules [4]. The DCP rules require the problem objectives to have one of two forms `minimize(convex)` and `maximize(concave)`. The only valid constraints under the DCP rules are `affine` == `affine`, `convex` <= `concave` and `concave` >= `convex`. In this project, we have used the `dccp` API, an extension of `CVXPY`, that relaxes some of the restrictions on problem constraints. The constraints in `dccp` can take the general form `l_i(x)` $\sim$ `r_i(x), i=1,...,m` where `l_i(x)` and `r_i(x)` are expressions with curvature known from the curvature composition rules and $\sim$ denotes one of the relational operators ==, <= or >=. However, problems with non-convex objectives or undefined-curvatures in constraints are still infeasible in `CVXPY`.

5.2 Supported Solvers

Among all modelling packages, `Pyomo` has the most extensive solver supports. From the documentation, Pyomo uses "solver managers" to execute solvers that perform optimization and other forms of model analysis. By default, Pyomo uses the serial solver manager to execute solvers locally. The serial solver managers support around 25 commercial and open-source solvers. Details can be found by installing `Pyomo` and typing `pyomo help --solvers` to the command line. 13 solvers are supported in `CVXPY` (link). It is a smaller set than `Pyomo` but does cover most convex optimization problems. Solvers supported by `Scipy` can be found following this link. Many solvers are pretty outdated with a history as old as `Scipy` itself and are no longer maintained. The `GUROBI` interface does not interact with solvers other than the `GUROBI` solver itself.

5.3 Ease of Use

The process of problem formulation in `Pyomo` and `GUROBI` are, in general, more tedious than that in the other packages. The two packages only allow element-wise operations, usually making the expressions long and hard to read (whereas all other packages support linear algebra). Moreover,

the two packages do not allow conditional statements in the problem definitions to depend on model variables, making it difficult for us to distinguish between the bid and ask costs. Therefore, when using the two packages, we have introduced auxiliary variables to represent the positive and negative changes in weights. Additional constraints are necessary to relate the auxiliary variables to the weights before and after rebalancing.

As a final remark, we note that the original experimental setup also includes the `NLopt` and `GEKKO` modelling suites. However, it was evident from an early stage that they have considerably worse efficiencies compared to the other packages. We have subsequently excluded them from further evaluations.

6 Conclusion

In sum, we recommend `(Pyomo, IPOPT)` for solving Markowitz' mean-variance portfolio optimization models with convex quadratic objectives and complex non-linear constraints. Pyomo is a flexible modelling package that allows very generic problem formulation and supports a diverse set of commercial and open-sourced solvers. IPOPT is a powerful open-sourced solver that can efficiently tackle large-scale non-linear problems.

Limitations

In hindsight, the author recognizes three weaknesses of this investigation:

Combined Evaluation: In the investigation, we have tested the efficiencies of the modelling package + solver pairs combined. However, we are also interested in the solver efficiencies independently, in which case we could prepare our problems in the `.nl` format (a standard format for presenting and archiving mathematical programming problems) and send them directly to the solvers. We might also be interested in separately testing the modelling packages with the same solver.

Curvatures of Constraints: In the budget constraints, we have defined our trading cost functions to have known curvatures (by the controlling the signs of various terms), in order to satisfy the `dccp` rules in CVXPY and standardize the comparisons across all combinations. However, the recommended `(Pyomo, IPOPT)` does have the capability to deal with more generic constraints and non-convex objective functions. The performance of `IPOPT` on these types of problems should be investigated further.

Different Problem Formulations: The problems formulated in Pyomo and GUROBI are slightly different from the that in other packages, although with the same expected optimal results. We have introduced the auxiliary variables to circumvent the inclusion of conditional statements in the constraints unsupported by the two packages. We could standardize the formulations more carefully.

Further Work

PySCIPOpt: In addition to `IPOPT`, another open-sourced non-linear solver that receives good reputation is `SCIP`. However, its performance is relatively poor according to our experiments. This might be due to the ineptitude of the solver itself but may also come from a potentially bad compatibility of `SCIP` in `Pyomo`. There is an actively maintained Python interface `PySCIPOpt` built

specifically for `SCIP` which receives many stars on GitHub. We are interested in evaluating this interface in the future.

Cloud Services: This investigation has focused on the solver performance when running on a local machine. However, there are cloud services (e.g. NEOS, OpenOpt, IBM Decision Optimization on Cloud) that allow remote solver access and are designed particularly for large-scale optimization problems. We are interested in testing their performance as well.

Dynamic Portfolio Optimization: In this investigation, we have limited our attention to the optimization model where the assets are allocated based on static expectations of investment returns. However, there have been interests in dynamic portfolio theories where we want to determine the optimal trading trajectory for a portfolio over a period of time. The role played by trading costs would be different in this approach. For example, frequent portfolio rebalancing that introduces large trading costs over time is likely to be penalized [5].

References

[1] Andersen, Martin S., et al. "CVXOPT ." CVXOPT, 17 Apr. 2020, cvxopt.org/.

[2] A. Wächter. An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, January 2002.

[3] A. Wächter and L. T. Biegler, On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, Mathematical Programming 106(1), pp. 25-57, 2006

[4] Boyd, Stephen P. "Disciplined Convex Programming." Home, 2020, dcp.stanford.edu/.

[5] Brown, David B., and James E. Smith. "Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds." Management Science 57.10 (2011): 1752-1770.

[6] "Chapter 4 Convex Optimization Problems." Convex Optimization, by Stephen P. Boyd and Lieven Vandenberghe, Cambridge Univ. Pr., 2011, pp. 127–213.

[7] Chen, Pierre, et al. "A Note on Portfolio Optimization with Quadratic Transaction Costs." SSRN Electronic Journal, 2019, doi:10.2139/ssrn.3683466.

[8] Ferreau, Hans Joachim, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. "qpOASES: A parametric active-set algorithm for quadratic programming." Mathematical Programming Computation 6, no. 4 (2014): 327-363.

[9] FICO Xpress Optimization. "Solution Methods." FICO, 2019, www.fico.com/fico-xpress-optimization/docs/latest/solver/optimizer/HTML/chapter4.html.

[10] Frazzini, Andrea, Ronen Israel, and Tobias J. Moskowitz. "Trading costs of asset pricing anomalies." Fama-Miller working paper (2012): 14-05.

[11] Frazzini, Andrea, et al. "Trading Costs." SSRN Electronic Journal, 2018, doi:10.2139/ssrn.3229719.

[12] Gurobi Optimization. "Advanced Gurobi Algorithms." Gurobi Optimization, 2016, www.gurobi.com/pdfs/user-events/2016-frankfurt/Die-Algorithmen.pdf.

[13] Jeyakumar, V., and G. Li. "Exact Second-Order Cone Programming Relaxations for Some Nonconvex Minimax Quadratic Optimization Problems." SIAM Journal on Optimization, vol. 28, no. 1, 2018, pp. 760–787., doi:10.1137/16m1058480.

[14] J. Nocedal, A. Wächter, and R.A. Waltz. Adaptive barrier strategies for nonlinear interior methods. SIAM Journal on Optimization, 19(4):1674–1693, 2008.

[15] Kolm, Petter N., et al. "60 Years of Portfolio Optimization: Practical Challenges and Current Trends." European Journal of Operational Research, vol. 234, no. 2, 2014, pp. 356–371., doi:10.1016/j.ejor.2013.10.060.

[16] Lobo, Miguel Sousa, et al. "Applications of Second-Order Cone Programming." Linear Algebra and Its Applications, vol. 284, no. 1-3, 1998, pp. 193–228., doi:10.1016/s0024-3795(98)10032-0.

[17] Lobo, Miguel Sousa, et al. "Portfolio Optimization with Linear and Fixed Transaction Costs." Annals of Operations Research, vol. 152, no. 1, 2006, pp. 341–365., doi:10.1007/s10479-006-0145-1.

[18] Markowitz, Harry. "Portfolio Selection." The Journal of Finance, vol. 7, no. 1, 1952, p. 77., doi:10.2307/2975974.

[19] MOSEK ApS. "Introducing the MOSEK Optimization Suite 9.2.29." MOSEK Optimization Suite, 28 Oct. 2020, docs.mosek.com/9.2/intro/index.html.

[20] Shen, Xinyue, et al. "Disciplined Convex-Concave Programming." 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, doi:10.1109/cdc.2016.7798400.

[21] Stellato, Bartolomeo, et al. "OSQP: an Operator Splitting Solver for Quadratic Programs." Mathematical Programming Computation, vol. 12, no. 4, 2020, pp. 637–672., doi:10.1007/s12532-020-00179-2.

[22] Vigerske, Stefan. "Coin-or/Ipopt." GitHub, 2020, github.com/coin-or/Ipopt.

A Precision of Optimal Results

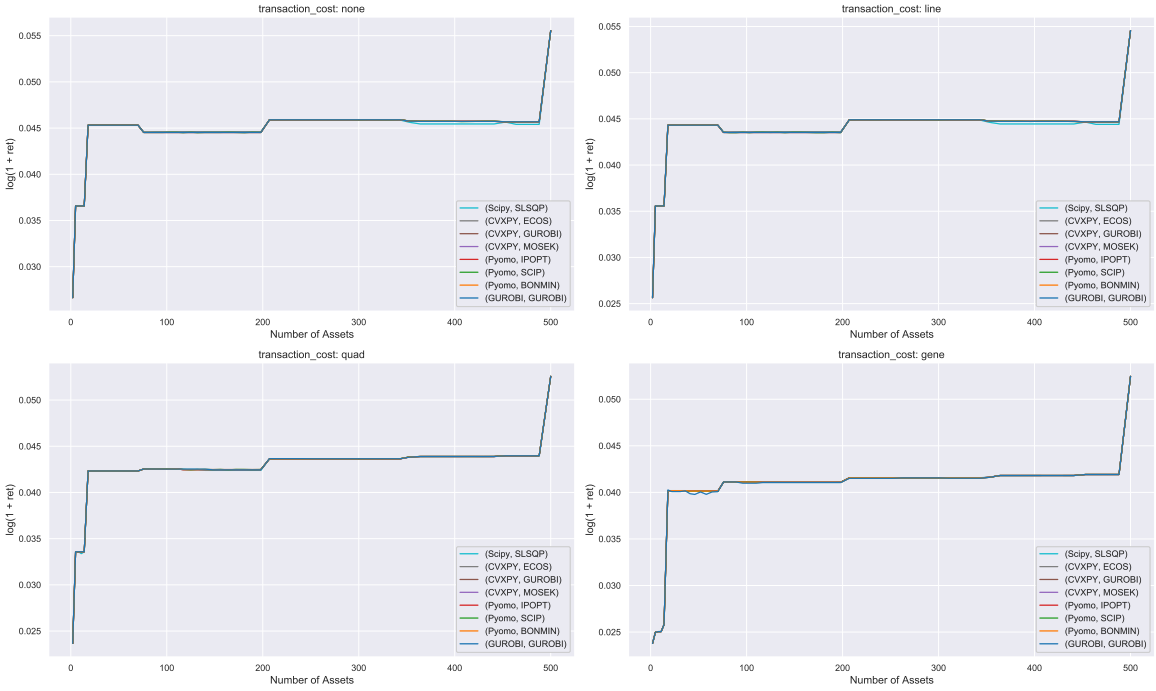

Figure 4: Optimal solutions from test (a). The optimal solutions from different solvers overlap for all type of costs, demonstrating the high precision consistency among the solvers.

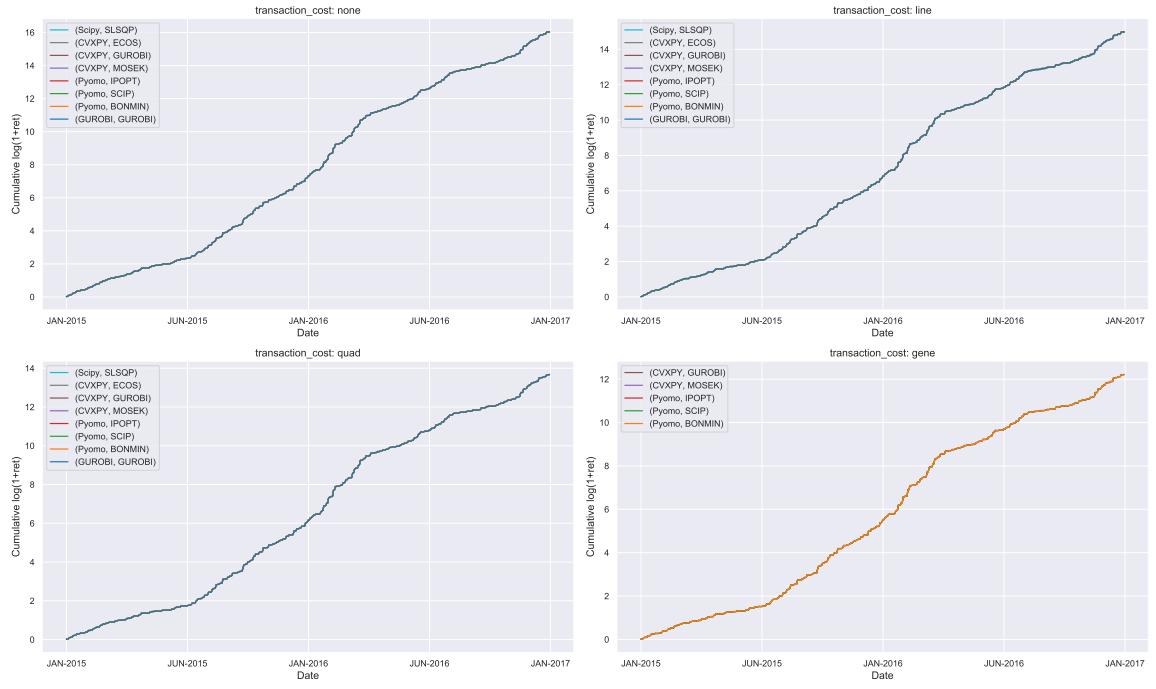

Figure 5: Cumulative returns from test (b). The cumulative returns from different solvers overlap for all type of costs, demonstrating the high consistency of precision among the solvers.

B An Alternative Formulation

The vector of portfolio weights w in the problem defined in Section 1 does not represent the actual allocation that can be traded. However, we note that the form in Section 1 can be considered as a reformulation of an equivalent problem with respect to the vector of tradable allocation $\hat{w}$ with budget constraint $\mathbf{1}_n^\top \hat{w} = 1$. In this alternative formulation with respect to $\hat{w}$, the vector of trading costs $c\left(\hat{w} \mid \tilde{w}\right)$ and the total trading cost $\mathcal{C}\left(\hat{w} \mid \tilde{w}\right) = \mathbf{1}_n^\top c\left(\hat{w} \mid \tilde{w}\right)$ are expressed as functions of $\hat{w}$ and $\tilde{w}$. Trading costs have two effects on our expected return a) direct expenses incurred when buying and selling the assets b) loss of expected return due to the reduced amount of wealth effectively allocated to the assets. It follows that the optimization problem can be written as

$$\begin{aligned} \underset{\hat{w} \in \mathcal{R}^N}{\text{minimize}} \quad & -\left[\left(\left(\hat{w} - c\left(\hat{w} \mid \tilde{w}\right)\right)^\top \mu - \mathcal{C}(\hat{w} \mid \tilde{w})\right) - \frac{1}{2}\left(\hat{w} - c\left(\hat{w} \mid \tilde{w}\right)\right)^\top \Sigma \left(\hat{w} - c\left(\hat{w} \mid \tilde{w}\right)\right)\right] \\ \text{s.t.} \quad & \begin{cases} \mathbf{1}_n^\top \hat{w} = 1 \\ \tilde{w} + \Delta\hat{w}^+ - \Delta\hat{w}^- = \hat{w} \\ \mathbf{0}_n \leq \hat{w} \leq \mathbf{1}_n \end{cases} \end{aligned}$$

However, the disadvantage of this formulation is apparent – the objective function does not have well-defined curvature due to the $c\left(\hat{w} \mid \tilde{w}\right)^\top \mu$ term in the return and the cross-terms in the variance. The objective function, however, can be made convex by a change of variable $w = h^{-1}\left(\hat{w}\right) = \hat{w} - c\left(\hat{w} \mid \tilde{w}\right)$ and an approximation $c\left(\hat{w} \mid \tilde{w}\right) \approx c\left(w \mid \tilde{w}\right)$.[6] The objective function then becomes

$$\underset{w \in \mathcal{R}^N}{\text{minimize}} \quad -\left[\left(w^\top \mu - \mathcal{C}(w \mid \tilde{w})\right) - \frac{1}{2} w^\top \Sigma w\right]$$

The budget constraint can be similarly rewritten as $\mathbf{1}_n^\top \hat{w} = \mathbf{1}_n^\top \left(w + c\left(\hat{w} \mid \tilde{w}\right)\right) = \mathbf{1}_n^\top w + \mathcal{C}\left(\hat{w} \mid \tilde{w}\right) \approx \mathbf{1}_n^\top w + \mathcal{C}\left(w \mid \tilde{w}\right)$. The turnover constraint can be rewritten by defining $\Delta w^- = \max\left(\tilde{w} - w, 0\right)$ and $\Delta w^+ = \max\left(w - \tilde{w}, 0\right)$. The long-only constraint automatically holds as the induced trading cost is always smaller than the reallocated wealth.

$$\begin{aligned} \text{s.t.} \quad & \begin{cases} \mathbf{1}_n^\top w + \mathcal{C}(w \mid \tilde{w}) = 1 \\ \tilde{w} + \Delta w^+ - \Delta w^- = w \\ \mathbf{0}_n \leq w \leq \mathbf{1}_n \end{cases} \end{aligned}$$

Thus we recover the expression in Section 1. The allocation $\hat{w}$ can be recovered from w by the relation $\hat{w} = w + c\left(w \mid \tilde{w}\right)$ where $\mathbf{1}_n^\top \hat{w} = \mathbf{1}_n^\top w + \mathcal{C}(w \mid \tilde{w}) = 1$ is satisfied. The problem defined with

[6] a) By construction, the transformation $h : \mathcal{R}^N \longrightarrow \mathcal{R}^N$ is one-to-one with image covering the problem domain $\mathcal{D}$ (i.e. $\mathcal{D} \subseteq h\left(\mathbf{dom}\ h\right)$). Hence, the optimization problem

$$\underset{w \in \mathcal{R}^N}{\text{minimize}} \quad f_0\left(h(w)\right) \qquad \text{s.t.} \quad \begin{cases} f_i\left(h(w)\right) \leq 0 & i = 1, \ldots, n \\ g_i\left(h(w)\right) = 0 & i = 1, \ldots, p \end{cases}$$

and

$$\underset{\hat{w} \in \mathcal{R}^N}{\text{minimize}} \quad f_0\left(\hat{w}\right) \qquad \text{s.t.} \quad \begin{cases} f_i\left(\hat{w}\right) \leq 0 & i = 1, \ldots, n \\ g_i\left(\hat{w}\right) = 0 & i = 1, \ldots, p \end{cases}$$

are equivalent – if w solves the first problem, then $\hat{w}$ solves the second problem and vice versa [6].

b) Using the relation $w = \hat{w} - c\left(\hat{w} \mid \tilde{w}\right)$, the trading cost vector $c\left(w \mid \tilde{w}\right) = c\left(\hat{w} - c\left(\hat{w} \mid \tilde{w}\right) \mid \tilde{w}\right)$. For small cost, we claim that $c\left(w \mid \tilde{w}\right) = c\left(\hat{w} \mid \tilde{w}\right) + \mathcal{O}\left(c^2\left(\hat{w} \mid \tilde{w}\right)\right)$. For example, in the case of linear trading cost $c\left(\hat{w} \mid \tilde{w}\right) = \kappa|\hat{w} - \tilde{w}|$, we have $c\left(w \mid \tilde{w}\right) = \kappa|\hat{w} - c\left(\hat{w} \mid \tilde{w}\right) - \tilde{w}| = \kappa|\hat{w} - \kappa|\hat{w} - \tilde{w}| - \tilde{w}| = c\left(\hat{w} \mid \tilde{w}\right) + \mathcal{O}\left(\kappa^2|\hat{w} - \tilde{w}|\right)$

respect to w and with trading cost $\mathcal{C}(w \mid \tilde{w})$ placed in the budget constraint has the advantage of making the objective function convex. The tradable allocation $\hat{w}$ can be recovered by adding the trading cost vector $c(w \mid \tilde{w})$, expressed as percentages of the total wealth incurred by rebalancing the portfolio, back to the effective allocation w.