THE UNIVERSITY OF BRITISH COLUMBIA

MATH 441

MATHEMATICAL MODELLING: DISCRETE OPTIMIZATION PROBLEMS

Optimizing Student Investment Experience

Submitted to
Patrick WALLS

Submitted by
Renée CHOQUETTE - 63948954
Henry CHUNG - 65685331
Theo DIEDERICHSEN - 86125424
Alex KHADRA - 99130759

May 10, 2024

Contents

List of Figures

Overview

This program aims to help students address the lack of resources for investment options by determining the optimal asset mix for maximum expected return, subject to their preferences. Using the constructed portfolio, this platform will provide an individual with the optimal brokerage selection while simultaneously minimizing the cost of investment fees. This program is intended to inform students on; where they want to invest their money, which method best satisfies their needs, and how they can maximize their returns while minimizing risk and cost.

Introduction

This document will lay out the computational and mathematical process taken to create a formula and program that will intake a user's investment experience, goals, and expectations, as well as their risk tolerance, sustainability requirements and intended effort, while providing them with the optimal path advised, in order to gain a maximum reward that aligns with their preferences. This project also aims to be a resource and educate the intended users of this program, and therefore before detailing the methods and processes of this project, the basics must be explained.

The general term for this problem, is a portfolio optimization problem. A portfolio, specifically an investment portfolio, is an individual's collection of assets [1]. There are different kinds of assets available that an individual can choose to include in their portfolio, according to their goals and preferences. This project focuses on financial assets, such as, stocks, bonds, mutual funds, Exchange-Traded Funds (ETFs), Guaranteed Investment Certificates (GICs), options, and cryptocurrencies.

Asset classes are the groups of assets that have similar characteristics and laws applications, for example, stocks and bonds are in different asset classes - equities and fixed income, respectively [4]. Investing involves buying stocks for a long-term gain, whereas trading is the term used for when individuals buy and sell stocks or shares for a short-term profit. They are bought and sold based on trends in the market [9].

Stocks – A share of a company, meaning that an individual who purchases a stock, owns a part of the company. Then when the company earns a profit, it shares part of the profit with owner's of their stocks, in dividends [2].

Bonds – A lower risk investment in the form of loaning money to a corporation, or the government, and in return for using an individual's capital, the company will pay back the loan with interest [3]. This makes a lower risk and lower reward form of investing for an individual [2].

Mutual Funds – When a portfolio manager organizes for several investors pool their money together to buy an asset mix. Several people invest together in several assets, and mutual funds are only traded once a day [5].

ETF – Similarly, ETFs are also a pooled investment fund, but they differ from mutual funds, as the share prices fluctuate as they are sold and traded throughout the day [6].

GIC – A secured investment, where an individual is guaranteed to get back the amount they invested after a predetermined period of time. GICs tend to have a long term deposit of at least one year[7].

Cryptocurrency – A form of digital exchange where there is no central authority that overlooks the value of cryptocurrency, instead it is maintained by it's users through the internet. It can be used buy goods, but mainly people use it as another form of investment, which requires extensive research as investing in cryptocurrency is high risk [8].

General Problem Form

The modern portfolio theory will be used to optimize portfolio allocation and brokerage selection. There will be N decision variable $x_1, \ldots, x_N$. The weight of each variable represents the portfolio weights (money per asset type). To ensure that the weights sum up to zero, multiply by $\mathbf{e}$, where $\mathbf{e}^T = \begin{bmatrix} 1 & \ldots & 1 \end{bmatrix}$, resluting in $\mathbf{e}^T\mathbf{x} = 1$.

The return on a portfolio can be represented as the sum of each of the weights multiplied by the return of each asset, $\mathbf{r}$, represented by $\sum_{i=1}^{N} r_i x_i = \mathbf{r}^T\mathbf{x}$. Similarly, the expected return can be represented as $\mathbb{E}[\mathbf{r}] = \boldsymbol{\mu}$, and the annualized expected return of the portfolio by $\sum_{i=1}^{N} \mu_i x_i = \boldsymbol{\mu}^T\mathbf{x}$.

Each asset mix has associated risk values. The risk will be measured by computing the variance of the total return of the portfolio. The portfolio's variance-covariance can be expressed as $\boldsymbol{\sigma}_{r_i}^2 = \mathbb{E}[r_i - \mathbb{E}[r_i]]^2 = \sum_i^N \sum_j^N x_i x_j \sigma_{ij}$. This can be represented in matrix notation form as $\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x}$, where $\boldsymbol{\Sigma}$ is the variance-covariance matrix. Based on the risk tolerance of the individual, they will be able to select a value of maximum acceptable risk, R, which will be the constraining variable.

The final constraint is that each asset would have a weight greater or equal to 0, which entails that the possibility of short-selling, will not be considered. This ensures that this analysis will be risk-averse for individuals.

This gives the generic optimization problem in the form of:

$$
\begin{aligned}
\max_{\mathbf{x}} \quad & \boldsymbol{\mu}^T\mathbf{x} \\
\text{s.t.} \quad & \mathbf{e}^T\mathbf{x} = 1 \\
& \mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x} \leq \mathbf{R} \\
& \mathbf{x} \geq 0
\end{aligned}
$$

Determining the Optimal Brokerage

There are many brokerages to chose from in Canada, each with slightly different fees and services provided. Ten of the largest and most common brokerages will be considered, to represent the most popular choices amongst investors. For each brokerage, a set of fees have been gathered such as, commissions and management fees for each asset type traded on the platform. If the asset type is not supported, the fee will be set to a very large number in order to ensure that brokerage is not selected if that service is required by the user.

Each of these fees diminish the return of a portfolio, therefore the best way to model the optimization, is to subtract the fees from the return of each asset type. It is noted that attempting to determine the optimal brokerage by using independent problems after having previously determined the asset mix, does not include the different fees for various asset types. This would mean the portfolio optimization problem would outweigh certain assets, like mutual funds, which may have a higher return before the deduction of fees but would be less appealing to investors when accounted for.

Another consideration is to formulate the optimal brokerage selection before determining an optimal asset mix. However this ignores the fact that some individuals could have varying risk tolerance, which may result in a vastly different portfolio selection than others. It would also render the problem redundant, as this process would return the same brokerage every time if the underlying assets are not known.

Therefore the adapted approach chosen will be to consider the various offers available at each brokerage. In order to find the best suited price for each investor, all different categories offered will be included in the problem. This way, each brokerage can be represented as a matrix object and the corresponding fee will be selected based on the input given by the investor prior to running the optimization problem.

The problem is represented in the following form:

$$\begin{aligned} \max_{\mathbf{x}} \quad & (\boldsymbol{\mu} - \mathbf{f})^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{e}^T \mathbf{x} = 1 \\ & \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} \leq \mathbf{R} \\ & \mathbf{x} \geq 0 \end{aligned}$$

Where $\mathbf{f}$ is the associated fee in percentage points associated to each asset type. Since $\mathbf{f}$ fluctuates depending on the brokerage, there will be different objective function per brokerage and separate problems will be used to compare each brokerage. The optimal result will be the highest expected return.

Problem Set Up

Portfolio Optimization Simplified Demonstration

In order to demonstrate the outworking of the aforementioned strategy, a simple example will be outlined as follows.

Assume that only one brokerage is used for all of the investments and the acceptable risk measure determined for the portfolio is, 0.003[1]. Here are three asset types that will be considered:

- Stock A with expected return of 5% denoted by μ_A
- Bond B with expected return of 4% denoted by μ_B
- ETF C with expected return of 8% denoted by μ_C

To create a realistic variance-covariance matrix, assumptions about the covariance between the returns of the assets need to be made. Assume the following historical returns per asset were gathered to calculate the variance of the assets' returns.

Assume the following variance values for each asset:

- Stock A with variance of return Var$(\mu_A) = (5\%)^2 = 0.0025$
- Bond B with variance of return Var$(\mu_B) = (4\%)^2 = 0.0016$
- ETF C with variance of return Var$(\mu_C) = (8\%)^2 = 0.0064$

Assume the following correlation coefficients between the returns of the assets:

- Stock A with Bond B Corr$(\mu_A, \mu_B) = 0.3$
- Bond B with ETF C Corr$(\mu_B, \mu_C) = 0.5$

[1]This is arbitrary. The proper method used to translate an individuals acceptable risk measure into a numerical value will be discussed later in this proposal document. This is a random value used to for the sake of demonstrating the example.

- ETF C with stock A $\text{Corr}(\mu_C, \mu_A) = 0.4$

The covariance between each pair of assets is then calculated using the following formula:

$$\text{Cov}(\mu_i, \mu_j) = \text{Corr}(\mu_i, \mu_j) \times \text{StdDev}(\mu_i) \times \text{StdDev}(\mu_j)$$

Now, the covariance values are calculated:

$$\begin{aligned}
\text{Cov}(\mu_A, \mu_B) &= 0.3 \times \sqrt{0.0025} \times \sqrt{0.0016} = 0.0006 \\
\text{Cov}(\mu_B, \mu_C) &= 0.4 \times \sqrt{0.0016} \times \sqrt{0.0064} = 0.00128 \\
\text{Cov}(\mu_A, \mu_C) &= 0.5 \times \sqrt{0.0025} \times \sqrt{0.0064} = 0.002
\end{aligned}$$

Using these results, a variance-covariance matrix is constructed. Let's denote the matrix as Σ:

$$\Sigma = \begin{bmatrix} 0.0025 & 0.0006 & 0.002 \\ 0.0006 & 0.0016 & 0.00128 \\ 0.002 & 0.00128 & 0.0064 \end{bmatrix}$$

Each element in the matrix represents the variance of an asset's returns along the diagonal, and the covariance between the returns of two different assets off the diagonal.

To further this example, assume that two different brokerages are considered, Broker A and Broker B. They have annual fee schedules:

Broker	Stock Fee	Bond Fee	ETF Fee
Broker A	1%	2%	0%
Broker B	2%	1%	3 %

These values are inputted into the general form to have two separate optimization problems:

The first problem for Broker A:

$$\begin{aligned}
\max_{\mathbf{x}} \quad & (0.05 - 0.01)x_1 + (0.04 - 0.02)x_2 + (0.08 - 0.00)x_3 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 = 1 \\
& 0.0025x_1^2 + 0.0006x_1x_2 + 0.002x_1x_3 \\
& +0.0006x_1x_2 + 0.0016x_2^2 + 0.00128x_2x_3 \\
& +0.002x_1x_3 + 0.00128x_2x_3 + 0.0064x_3^2 \leq 0.003 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}$$

And the second problem for Broker B:

$$
\begin{aligned}
\max_{\mathbf{x}} \quad & (0.05-0.02)x_1 + (0.04-0.01)x_2 + (0.08-0.03)x_3 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 = 1 \\
& 0.0025x_1^2 + 0.0006x_1x_2 + 0.002x_1x_3 \\
& +0.0006x_1x_2 + 0.0016x_2^2 + 0.00128x_2x_3 \\
& +0.002x_1x_3 + 0.00128x_2x_3 + 0.0064x_3^2 \leq 0.003 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
$$

Running these problems in python using, **cvxpy**, the results show that the best return for Broker A is approximately 5.8% after fees and the allocation is 40% in Stock A, 10% in Bond B and 50% in ETF C.

The best return for Broker B is approximately 4.1% after fees and the allocation is 7% in Stock A, 42% in Bond B and 57% in ETF C.

Hence, this determines that the optimal portfolio mix for this problem is 40% in Stock A, 10% in Bond B and 50% in ETF C and to go with Broker A (refer to Appendix A for the python code).

Identifying Constraints

- The portfolio as well as the individual assets, must have a maximum constraint
- Brokerage asset limitations
- Varying brokerage offerings for students, young people, active traders, etc
- Trading frequency
- Initial investment amount
- User's expected time span of investment
- Local investments
- Risk tolerance

As different brokerages are considered, not all of them offer all investment products. For example, Equitable Bank (EQ) offers comparably better rates for GICs [1], however that is their sole investment product offered. Therefore, some asset types may not considered for specific optimization problems. The most effective way to represent this in the objective function is to set the fee to a very large value[2] for that brokerage. This allows the asset to be excluded from the optimization problem.

Most brokerages do not provide their fee schedule in terms of year over year percentage basis, but rather in a dollar value per trade[3]. This fee will be translated into a year over year (annually compounded) percentage point basis to be compatible with our objective function. By surveying how frequently the individual wants to manage their portfolio, it is possible to determine the number of trades they are likely to place over a year. By multiplying the fee with the number of trades per year, and dividing this by the total portfolio value, the corresponding percentage for year over year fee will be given.

[2] $10^t p$, ten to the power of investment time period multiplied by the portfolio value, is chosen for this, as there is no asset type with a return near 10^t%.

[3] Also know as commission.

Brokerages also offer special pricing for some individuals. For example, CIBC offers stock trading at \$6.95 per trade for regular traders, \$5.95 per trade for students, \$4.95 per trade for active investors, and \$0.00 per trade for young people [2]. This means the rate provided by a brokerage can be tailored to the individual's specifications. These rates for each brokereage will be represented as matrix objects, with each row representing fees for each investor type. An important observation to note, is that active traders receive the best rate, followed by young investors, then student investors, and finally regular investors. This is consistent with the fees found across all Canadian brokerages.

General fees differ according to the brokerage, such as opening fees, and activity fees. These fees will only be considered if they are recurrent, as this problem is intended to determine the return over a long period of time. When the fees are recurrent, they will be included as a negation of the term **b**. Similarly, if this value is not represented in a year over year percentage basis, then it will be converted by dividing the fee by the total portfolio value.

In the prior example the investment was limited to only one brokerage, in order to keep track of the investments. Yet, here are individuals that may want to have multiple accounts open. This simplifies the problem by selecting the lowest fee from each brokerage and combining those values into one single problem. The result of that problem will give the optimal value and asset allocation weights.

Various constraints will be selected, such as minimum ESG *(Environmental, Sustainable, and Governance)* score. ESG scores are a tool used by investors, so they can assess a company's ethical performance. These scores range from 0 to 100, with a poor score being below 50, and a good score being above 70 [3]. In this project, the user will set a minimum ESG score required to best suit their standards.

With this additional constraint, the problem is as follows:

$$\begin{aligned} \max_{\mathbf{x}} \quad & (\boldsymbol{\mu} - \mathbf{f})^T \mathbf{x} - \mathbf{b} \\ \text{s.t.} \quad & \mathbf{e}^T \mathbf{x} = 1 \\ & \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} \leq \mathbf{R} \\ & \boldsymbol{\epsilon}^T \mathbf{x} \geq \boldsymbol{\varepsilon} \\ & \mathbf{x} \geq 0 \end{aligned}$$

All the mentioned constraints from above can be added to the optimization problem.

Portfolio Optimization Demonstration Refined

This example will revisit the first example, by adding the above constraints. In this example the user is a student investor, does not trade frequently, wants to invest \$10,000 over a one year period, would like to keep their money in one place (singular brokerage), they are fairly environmentally conscious and would like a minimum ESG score of 75, but they do not care much about risk, therefore the determined acceptable risk measure is 0.005.

The asset options are the same as in the previous example, with additional ESG scores.

- Stock A with expected return of 5% denoted by μ_A and ESG score of 75
- Bond B with expected return of 4% denoted by μ_B and ESG score of 80
- ETF C with expected return of 8% denoted by μ_C and ESG score of 65

Along with the same variance covariance matrix of:

$$\Sigma = \begin{bmatrix} 0.0025 & 0.0006 & 0.002 \\ 0.0006 & 0.0016 & 0.00128 \\ 0.002 & 0.00128 & 0.0064 \end{bmatrix}$$

However, the brokerages now have different fees for different individuals, along with a new broker, Broker C, which does not offer bond trading. They are as follows:

Broker A	Stock Fee	Bond Fee	ETF Fee
Regular Investor	1%	2%	0%
Student Investor	1%	2%	0%
Active Investor	1%	2%	0%
Young Investor	1%	2%	0%

Broker B	Stock Fee	Bond Fee	ETF Fee
Regular Investor	2%	1%	3%
Student Investor	1%	1%	1%
Active Investor	1%	0%	0%
Young Investor	1%	0%	0%

Broker C	Stock Fee	Bond Fee	ETF Fee
Regular Investor	$4.95 per trade	-	$4.95 per trade
Student Investor	$4.95 per trade	-	$2.95 per trade
Active Investor	$0 per trade	-	$0 per trade
Young Investor	$4.95 per trade	-	$2.95 per trade

It is assumed that Broker C has a yearly activity fee of 1%.

To convert the dollar-value of fees into a year-compounding percentage we define the following terms:

- Frequency of Trading (f): The frequency of trading represents how often the investor buys or sells assets within a year [4]. In this case, it is assumed that "not actively trading" means the investor only executes trades when buying and selling assets for a one-year time frame. The assumption has also been made that "active trading" is defined as a threshold of 600 trades per year.
- Term Length (t): The term length refers to the duration for which the investor holds the assets before selling them. In this scenario, it's assumed to be one year, corresponding to the one-year time frame mentioned earlier.
- Total Asset Value (p): This is the total value of the investor's assets.
- Fee Matrix for Broker C (C): Broker C charges fees for each trade, but the fees are expressed as dollar values rather than percentages. To make the fee calculation compatible with the optimization problem, the dollar-based fees are converted into a percentage of the total asset value. This conversion allows all fees to operate as percentages, making them directly comparable in the optimization problem.

[4] This is also arbitrary. Some investors who say they do not trade actively might still place trades more frequently than that, so it is assumed that active trading means 150+ trades per quarter, and not frequently to be only buying and holding until end of term. Anything in between will be a moderately frequent trader.

- Matrix of Ones (X): This matrix has the same dimensions as the fee matrix C and contains ones in every position. It's used to represent the fact that there is a fee associated with each asset.

The formula calculates the effective fee rate for each asset by adding a proportion of the total asset value to the fees charged by Broker C. This adjustment ensures that the fees are scaled appropriately relative to the total asset value. The term $\left(\frac{2+f\cdot t}{p}C+X\right)^{\frac{1}{t}}$ represents the effective fee rate matrix, where $\frac{2+f\cdot t}{p}$ scales the fees by the total asset value and the trading frequency, and raising this matrix to the power of $\frac{1}{t}$ adjusts it for the term length. Subtracting the matrix of ones (X) ensures that assets without fees remain unchanged. This formula computes the effective fee rate for each asset, considering the total asset value, trading frequency, term length, and the specific fee structure of Broker C. It allows for a more accurate representation of the fees incurred by the investor when optimizing the portfolio.

$$\left(\frac{2+f\cdot t}{p}C+X\right)^{\frac{1}{t}}-X$$

$$=\left(\frac{2+0(1)}{10000}\begin{bmatrix}4.95 & 10^t p & 4.95\\ 4.95 & 10^t p & 2.95\\ 0 & 10^t p & 0\\ 4.95 & 10^t p & 2.95\end{bmatrix}+\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\\1&1&1\end{bmatrix}\right)^{\frac{1}{1}}-\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$\approx\begin{bmatrix}0.1\% & 2000\% & 0.1\%\\ 0.1\% & 2000\% & 0.06\%\\ 0\% & 2000\% & 0\%\\ 0.1\% & 2000\% & 0.06\%\end{bmatrix}$$

Broker	Stock Fee	Bond Fee	ETF Fee
Broker A	1%	2%	0%
Broker B	1%	1%	1 %
Broker C	0.1%	2000%	0.6 %

Now, this is applied to the general form to get three separate optimization problems:

The first problem for Broker A:

$$\begin{aligned}\max_{\mathbf{x}} \quad & (0.05-0.01)x_1+(0.04-0.02)x_2+(0.08-0.00)x_3-0.00\\ \text{s.t.} \quad & x_1+x_2+x_3=1\\ & 0.0025x_1^2+0.0006x_1x_2+0.002x_1x_3\\ & +0.0006x_1x_2+0.0016x_2^2+0.00128x_2x_3\\ & +0.002x_1x_3+0.00128x_2x_3+0.0064x_3^2\leq 0.005\\ & 75x_1+80x_2+65x_3\geq 75\\ & x_1,x_2,x_3\geq 0\end{aligned}$$

The second problem for Broker B:

$$
\begin{aligned}
\max_{\mathbf{x}} \quad & (0.05-0.01)x_1 + (0.04-0.01)x_2 + (0.08-0.01)x_3 - 0.00 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 = 1 \\
& 0.0025x_1^2 + 0.0006x_1x_2 + 0.002x_1x_3 \\
& +0.0006x_1x_2 + 0.0016x_2^2 + 0.00128x_2x_3 \\
& +0.002x_1x_3 + 0.00128x_2x_3 + 0.0064x_3^2 \leq 0.005 \\
& 75x_1 + 80x_2 + 65x_3 \geq 75 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
$$

And the third problem for Broker C:

$$
\begin{aligned}
\max_{\mathbf{x}} \quad & (0.05-0.001)x_1 + (0.04-10.00)x_2 + (0.08-0.0006)x_3 - 0.01 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 = 1 \\
& 0.0025x_1^2 + 0.0006x_1x_2 + 0.002x_1x_3 \\
& +0.0006x_1x_2 + 0.0016x_2^2 + 0.00128x_2x_3 \\
& +0.002x_1x_3 + 0.00128x_2x_3 + 0.0064x_3^2 \leq 0.005 \\
& 75x_1 + 80x_2 + 65x_3 \geq 75 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
$$

Running these problems in python using, **cvxpy** the results show that the best return for Broker A is approximately 4.0% after fees and the allocation is 29% in Stock A, 47% in Bond B and 24% in ETF C.

The best return for Broker B is approximatley 4.3% after fees and the allocation is 0% in Stock A, 67% in Bond B and 33% in ETF C.

The best return for Broker C is approximatley 3.9% after fees and the allocation is 100% in Stock A, 0% in Bond B and 0% in ETF C.

Hence, it is determined that the optimal portfolio mix for this problem is 0% in Stock A, 67% in Bond B and 33% in ETF C, and to go with Broker B.

Portfolio Optimization Demonstration Finalized

Finally, the same values for the brokers and asset types are used, but characteristics of the user changes. The user now wants a lower risk tolerance and is satisfied with a value of 0.003, however they accept a minimum ESG score of 70. They are now an active investor, placing more than 600 trades per year. The total amount of money they wish to invest is $100,000 and the time horizon is 10 years.

With this new set of assumptions, the fee calculation for Broker C will be revisited. By applying the same procedure:

$$\left(\frac{2+f\cdot t}{p}C+X\right)^{\frac{1}{t}}-X$$

$$=\left(\frac{2+600(10)}{100000}\begin{bmatrix}4.95 & 10^t p & 4.95\\ 4.95 & 10^t p & 2.95\\ 0 & 10^t p & 0\\ 4.95 & 10^t p & 2.95\end{bmatrix}+\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\\1&1&1\end{bmatrix}\right)^{\frac{1}{1}}-\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$\approx\begin{bmatrix}2.63\% & 2287\% & 2.63\%\\ 2.63\% & 2287\% & 1.64\%\\ 0\% & 2287\% & 0\%\\ 2.63\% & 2287\% & 1.64\%\end{bmatrix}$$

However, since it is determined that using more than one broker, a simple fee schedule of 0% for stock fee is left if Broker C is chosen, 0% for bond fee if Broker B is chosen, and 0% ETF fee if Broker C is chosen. A 1% fee for using Broker C is also added. Hence, it returns one problem in the form of:

$$\begin{aligned}\max_{\mathbf{x}}\quad & (0.05-0.00)x_1+(0.04-0.00)x_2+(0.08-0.00)x_3-0.01\\ \text{s.t.}\quad & x_1+x_2+x_3=1\\ & 0.0025x_1^2+0.0006x_1x_2+0.002x_1x_3\\ & +0.0006x_1x_2+0.0016x_2^2+0.00128x_2x_3\\ & +0.002x_1x_3+0.00128x_2x_3+0.0064x_3^2\leq 0.003\\ & 75x_1+80x_2+65x_3\geq 70\\ & x_1,x_2,x_3\geq 0\end{aligned}$$

If broker C omitted due to the fee, then there is a problem in the form of:

$$\begin{aligned}\max_{\mathbf{x}}\quad & (0.05-0.01)x_1+(0.04-0.00)x_2+(0.08-0.00)x_3-0.00\\ \text{s.t.}\quad & x_1+x_2+x_3=1\\ & 0.0025x_1^2+0.0006x_1x_2+0.002x_1x_3\\ & +0.0006x_1x_2+0.0016x_2^2+0.00128x_2x_3\\ & +0.002x_1x_3+0.00128x_2x_3+0.0064x_3^2\leq 0.003\\ & 75x_1+80x_2+65x_3\geq 70\\ & x_1,x_2,x_3\geq 0\end{aligned}$$

Running these problem in python, the result using a combination of all three brokers is approximately a 5.4% return after fees, and the allocation is 29% in Stock A, 17% in Bond B, and 53% in ETF C.

Using the latter problem, the return is approximately 6.3%, with 0.7% in Stock A, 42% in Bond B, and 57% in ETF C.

Hence, the latter option is chosen and Broker C is not used due to the cost of the yearly fee.

Refer to Appendix B for the sample code which is used to compute these values.

Methods

Brokerage Selection

A function is employed to filter through the brokerages and select those meeting the user's specified criteria. Beginning with ten brokerages, the function uses a generic parameter determined by the user's input requirements, then it creates a filtered list of suitable brokers with the associated fees for each brokerage and investment category. This process streamlines the selection of brokerage options based on user-defined parameters to improve user decision-making. The function produces the optimal brokerage(s) selection with the highest return.

Decision Variable Selection

To formulate the optimization problem, various investment types commonly chosen by Canadian investors need to be considered [12]. These include GICs (for the sake of simplicity, only one-year term GICs will be considered), stocks, bonds, ETFs, and cryptocurrencies. Initially, the dataset used contained over 10,000 investment types. However, due to constraints posed by the data sources **yfinance** and **yesg**, the extensive dataset was filtered, resulting in under 500 investment types that were compatible with this program's analysis modules.

For reference, the dataset containing the decision variables, including the identified investment types, can be accessed via the CSV file named "tickers_database.csv". This dataset serves as the foundation for the optimization model, allowing considerations of a diverse range of investment options within the defined asset classes.

Just as with the process for selecting brokerages, a filtering function is used to narrow down which investment types are suitable for the user in the optimization problem. This function evaluates user-specified criteria, such as domestic investment preferences or carbon emission constraints, and selects only a subset of all available options that meets the criteria. By narrowing down the scope of available investment types, this approach simplifies the optimization problem, making it more computationally efficient and specific to the user's objectives.

Calculating Expected Return

For each investment type, the expected return was determined, using the maximum available historical data obtained. The entire list of filtered tickers was gathered and empty lists initialized to accumulate the net returns. This return was then transformed into a yearly-compounded rate, using the following formula:

$$\text{Expected Return} = \left(\frac{\text{Adjusted Value at End} - \text{Value at Start}}{\text{Value at Start}} + 1\right)^{\left(\frac{365}{\text{\# of Days between Start and End}}\right)} - 1$$

Where:

- *Expected Return* denotes the estimated expected return of the portfolio.
- *Adjusted Value at End* represents the portfolio value at the end of the specified time frame, inclusive of dividends.
- *Value at Start* signifies the initial value of the asset at the start of the available data.

- *# of Days between Start and End* denotes the count of days between the start index and the end index in the historical data.

This formula derives the expected return by analyzing the initial and final prices of the investment across a designated time span, factoring in compounding effects and dividends.

Given the distinct fee structures of each broker for various investment types, separate lists were curated corresponding to each broker. Consequently, diverse sets of expected returns for each broker were obtained, encompassing scenarios wherein different brokers are engaged for investment purposes. These lists serve as coefficients for the objective function in the subsequent optimization problem.

Refer to Appendix C for the python code associated with the method.

Overall, this function calculates the expected returns adjusted for broker fees for a variety of investment types and brokerages, facilitating decision-based investment for net expected returns.

The confidence level interval was determined on the rate of return of the portfolio, using the risk-level of the total asset mix. It is a way to represent the range of possible outcomes for the portfolio's growth over time, considering the uncertainty in the expected return rate.

Using the concept of standard deviation and assuming a normal distribution of returns, the square root of the total variance is used to find the standard deviation, thus the risk of the portfolio. Then, by applying a certain z-score a confidence interval is found by applying the formula:

$$\text{Confidence Interval} = \text{expected\_return} \pm Z \times \text{Volatility}$$

Where:

- Z is the critical value corresponding to the desired confidence level.
- Volatility is the standard error of the estimated expected return.

Refer to Appendix D for the python code of the process described above.

Variance-Covariance Calculation

Daily return data spanning a one-year period was used to construct the variance-covariance matrix. This dataset allowed assessments to be made on the correlation of various types of assets with one another. In portfolio optimization, where the aim is to distribute assets to achieve a desired risk-return balance, it's important that the variance-covariance matrix is semi-positive definite (SPD). This ensures that the resulting portfolio allocations are meaningful and coherent.

To ensure this, it was verified that the sum is SPD, meaning that for any non-zero vector of portfolio weights x, the quadratic form $x^T \Sigma x$ yields a non-negative result:

$$x^T \Sigma x \geq 0$$

Additionally, various risk metrics were examined, such as risk neutrality, aversion, or seeking behavior. A constraint was enforced to limit the overall risk level, R, of the portfolio of a specified threshold. This gave the following constraint:

$$x^T \Sigma x \leq R$$

Refer to Appendix E for the python code of the previous description.

ESG Computation

To ensure that each asset type in the portfolio had an ESG score, it was determined that when an asset type, such as GICs, did not have ESG scores, a minimum acceptable ESG score determined by the user, min_esg , replaced the blank value. This resulted in the constraint:

$$x^T E \geq min\_esg$$

Refer to Appendix F for the python code.

Results

Running the program, with minimal constraints and the following assumptions will give a general overview on how to interpret the results. The individual considered in this example has:

- No active trading on the portfolio (buy and leave it approach)
- A ten year time horizon
- $10,000 to invest
- Medium risk tolerance
- Medium level of ESG consciousness
- Student investor
- No preference for in-person advising
- Prefer to use only one brokerage
- No preference on local investment

The resulting portfolio weight as suggested by the program are demonstrated in figure 1

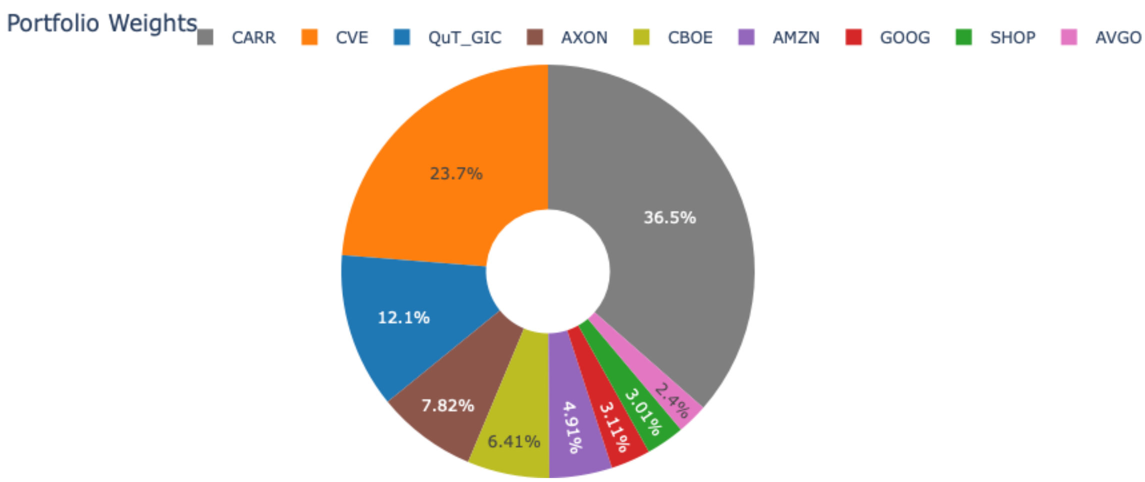

Figure 1: Suggested Portfolio Weights

The brokerage suggested in order to receive the maximal return is, Questrade. Using the portfolio volatility, the computed 95% confidence interval of the portfolio is between 12.4% and 16.4%. This is shown in figure 2.

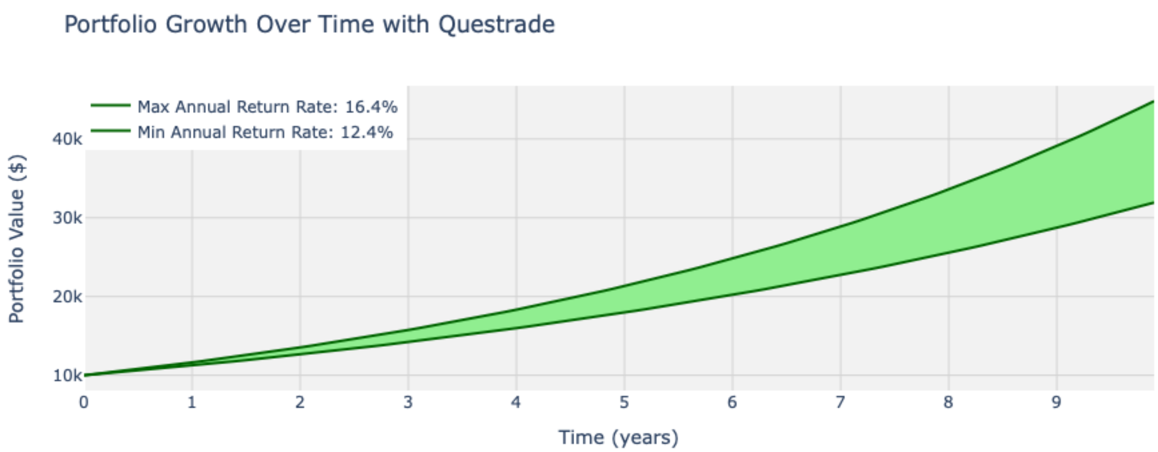

Figure 2: Projected returns with Questrade

Refer to Appendix G for the python code.

Limitations

While the project provides a great basis for portfolio optimization along with brokerage selection, there are important areas in which this project could be expanded. The main limitation was the lack of open source data available to work with. This made the optimization problem limited to a certain number of decision variables, which may not accurately represent real-life decision making. While various constraints were included to filter out investment options, there are also other important constraints which could better represent an individual's preferences. For example, a constraint determining whether an asset negatively impacts the environment, other than a low ESG score. However, this would require extensive research, which no program meant to best aid a general population would be able to encompass.

Accessibility to Open-Source Data

yfinance was used as the sole database to access stock data. While it has a relatively large amount of data, it is lacking compared to other sources, such as Bloomberg. Due to the inaccessibility of GIC data and mutual fund data, additional research had to be conducted to write GIC data in a format which could be utilized. Due to the variable nature of mutual fund returns, mutual funds were omitted as a decision variable.

Recently, **yfinance** also removed data for ESG scores, so the database **yesg** was used to determine ESG scores. Due to the limitations of the **yesg** database, the method mentioned previously of using user determined values where data was left blank was used, affecting the programs accuracy.

With access to a larger dataset, additional decision variables and including mutual funds as an investment option would further the diversification options to investors.

Computation of Expected Return

In this project, the computation of expected returns was simplified by relying on historical data and past returns were used to estimate the expected return of the assets. This method has limitations, as it only looks back to identify the best-performing asset over a specific time period.

In professional finance, a more robust approach involves incorporating forecasts and expectations from analysts and forecasters. By considering a weighted average of various predictions, and validating them through back-testing on multiple portfolios, a more accurate expected return for the portfolio can be computed. Access to databases such as Bloomberg would help obtaining analyst forecasts and predictions, improving the accuracy of our expected return calculations.

Integration with User Interface

Ideally, this system would seamlessly integrate with a user interface (UI), streamlining the process for users. This integration would facilitate the input of essential variables and present the results in a more organized manner. Theoretically, the implementation of this integration should be relatively straightforward, but this was not possible due to financial and time constraints. Business Intelligence tools such as Power BI and Tableau offer a straight forward way for users to select options (in this case constraints) and have python code running in the background to produce some output. These tools allow you to publish your dashboard as a web application.

Bibliography

References

[1] Curry, B. (2023, May 18). What is a portfolio?. Forbes.

[2] Investment portfolio. Corporate Finance Institute. (2023a, December 19).

[3] Chen, J. (2023, October 12). Investing for beginners: A guide to assets. Investopedia.

[4] Ganti, A. (2024, April 2). What are asset classes? more than just stocks and Bonds. Investopedia.

[5] Hayes, A. (2024, March 7). Mutual funds: Different types and how they are priced. Investopedia.

[6] Chen, J. (2024, April 12). Exchange-traded fund (ETF) explanation with pros and cons. Investopedia.

[7] Financial Consumer Agency of Canada. (2024, February 2). Government of Canada. Canada.ca.

[8] Ashford, K. (2024, April 10). What is cryptocurrency?. Forbes.

[9] Ashburn, D. (2022, December 13). Day trading, active trading, and investing: What's the difference? Encyclopædia Britannica.

[10] GICs. EQ Bank. (n.d.).

[11] CIBC. (n.d.). Our low pricing helps your money go further. Pricing — CIBC Investor's Edge.

[12] What is ESG and what does it mean for your business?. BDC.ca. (2024, February 12).

[13] TD. (n.d.). Common asset classes found in a portfolio. TD Direct Investing Education.

[14] Mains, N. E. (1977). Risk, the pricing of capital assets, and the evaluation of investment portfolios: Comment. The Journal of Business, 50(3), 371–384. https://doi.org/10.1086/295952

[15] Kenton, W. (2022, July 25). Historical returns: Definition, uses, and how it's calculated. Investopedia.

Appendices

Appendix A

```python
import cvxpy as cp
import numpy as np

brokerA = np.array([0.01, 0.02, 0.00])
brokerB = np.array([0.02, 0.01, 0.03])

mu_1 = np.array([0.05, 0.04, 0.08])
Sigma = np.array([[0.0025, 0.0006, 0.002],
              [0.0006, 0.0016, 0.00128],
              [0.002, 0.00128, 0.0064]])
f = brokerB
mu = mu_1 - f
R = 0.003
n = len(mu)
x = cp.Variable(n)


objective = cp.Maximize(mu.T @ x)
constraints = [cp.sum(x) == 1,
               cp.quad_form(x, Sigma) <= R,
               x >= 0]

problem = cp.Problem(objective, constraints)

problem.solve()

optimal_value = problem.value
optimal_solution = x.value
print("Optimal value:", optimal_value)
print("Optimal solution:", optimal_solution)
```

Appendix B

```
import cvxpy as cp
import numpy as np

fq = 600
t = 10
p = 100000
R = 0.003
min_esg = 70
inv_type = 2

brokerA_i = np.array([[0.01, 0.02, 0.00],
                   [0.01, 0.02, 0.00],
                   [0.01, 0.02, 0.00],
                   [0.01, 0.02, 0.00]])
brokerB_i = np.array([[0.02, 0.01, 0.03],
                   [0.01, 0.01, 0.01],
                   [0.01, 0.00, 0.00],
                   [0.01, 0.00, 0.00]])
C = np.array([[4.95, (10**t)*p ,4.95],
              [4.95, (10**t)*p, 2.95],
              [0, (10**t)*p ,0],
              [4.95,(10**t)*p , 2.95]])
brokerC_i = cal_fee(C)
brokerMin = np.minimum.reduce([brokerA, brokerB, brokerC])

def cal_fee(matrix):
    n, m = matrix.shape
    one_matrix = np.ones((n,m))
    new_matrix = (((2+fq*t)/p) * matrix + one_matrix)**(1/t) - one_matrix
    return new_matrix

brokers = [brokerA_i,brokerB_i,brokerC_i,brokerMin]

esg = np.array([75,80,65])
mu_1 = np.array([0.05, 0.04, 0.08])
Sigma = np.array([[0.0025, 0.0006, 0.002],
              [0.0006, 0.0016, 0.00128],
              [0.002, 0.00128, 0.0064]])

for b in brokers:
    f = b[inv_type]
    mu = mu_1 - f
    n = len(mu)
    x = cp.Variable(n)

    objective = cp.Maximize(mu.T @ x - 0.00)
```

```
    constraints = [cp.sum(x) == 1,
                   cp.quad_form(x, Sigma) <= R,
                   esg.T @ x >= min_esg,
                   x >= 0]

    problem = cp.Problem(objective, constraints)

    problem.solve()

    optimal_value = problem.value
    optimal_solution = x.value
    print("Optimal return for broker:", optimal_value)
    print("Optimal solution:", optimal_solution)
```

Appendix C

```
def cal_return_with_broker_fee(investments):
    mean_data = []
    final_mean = []
    for i in investments:
        if i == 'EQ_GIC':
            mean_return = 0.03
            broker_mean = []
            for b in brokers:
                broker_mean.append(mean_return-b[0])
            mean_data.extend(broker_mean)
        elif i == 'QuT_GIC':
            mean_return = 0.026
            broker_mean = []
            for b in brokers:
                broker_mean.append(mean_return-b[1])
            mean_data.extend(broker_mean)
        elif i == 'RBC_GIC':
            mean_return = 0.025
            broker_mean = []
            for b in brokers:
                broker_mean.append(mean_return-b[2])
            mean_data.extend(broker_mean)
        elif i == 'BMO_GIC':
            mean_return = 0.025
            broker_mean = []
            for b in brokers:
                broker_mean.append(mean_return-b[3])
            mean_data.extend(broker_mean)
        elif i == 'TD_GIC':
            mean_return = 0.025
            broker_mean = []
            for b in brokers:
                broker_mean.append(mean_return-b[4])
            mean_data.extend(broker_mean)
        elif i == 'CIBC_GIC':
            mean_return = 0.025
            broker_mean = []
            for b in brokers:
                broker_mean.append(mean_return-b[5])
            mean_data.extend(broker_mean)
        else:
            data_stock = pd.read_csv('stock_data.csv', sep=';', header=None)
            data_etf = pd.read_csv('ETF_data.csv', sep=';', header=None)
            data_crypto = pd.read_csv('crypto_data.csv', sep=';', header=None)
            data_bond = pd.read_csv('bond_data.csv', sep=';', header=None)
            companies = yf.download(i, period='max')['Adj Close']
```

```python
            data = companies.copy()
            days = (data.index[-1] - data.index[0]).days
            mean_return = (((data[-1] - data[0])/data[0] + 1)**(365/(days))) - 1
            #mean_return = (data[-1] - data[0])/data[0]

            broker_mean = []
            for b in brokers:
                if (data_stock == i).any().any():
                    broker_mean.append(mean_return - b[6])
                elif (data_bond == i).any().any():
                    broker_mean.append(mean_return - b[7])
                elif (data_etf == i).any().any():
                    broker_mean.append(mean_return - b[8])
                elif (data_crypto == i).any().any():
                    broker_mean.append(mean_return - b[10])
            mean_data.extend(broker_mean)

    for i in range(0, len(mean_data), len(brokers)):
        final_mean.append(mean_data[i:i+len(brokers)])

    final_return = list(map(list, zip(*final_mean)))
    return final_return
```

Appendix D

```
volatility = np.sqrt(np.round(optimal_solution,3).T @ Sigma_m @ np.round(optimal_solution,3))
confidence_level = 0.95
z_score = norm.ppf((1 + confidence_level) / 2)
max_returns = max(returns)*0.4

time = np.arange(0, t, 0.1)  # Time horizon
initial_investment = p  # Initial investment amount
min_annual_return_rate = max_returns - volatility*z_score  # Minimum annual return rate
max_annual_return_rate = max_returns + volatility*z_score  # Maximum annual return rate
```

Appendix E

```python
def cal_cov(investments):
    EQ_gic = pd.read_csv('EQ_gic.csv', sep=';', header=None)
    QuT_gic = pd.read_csv('QuT_gic.csv', sep=';', header=None)
    Four_gic = pd.read_csv('Four_gic.csv', sep=';', header=None)
    cov_data = []
    for i in investments:
        if i == 'EQ_GIC':
            EQ_gic_ = EQ_gic.pct_change().values[0:252]
            cov_data.append(EQ_gic_)
        elif i == 'QuT_GIC':
            QuT_gic_ = QuT_gic.pct_change().values[0:252]
            cov_data.append(QuT_gic_)
        elif i in ['RBC_GIC', 'BMO_GIC', 'TD_GIC', 'CIBC_GIC']:
            Four_gic_ = Four_gic.pct_change().values[0:252]
            cov_data.append(Four_gic_)
        else:
            companies = yf.download(i, period='1y')['Adj Close']
            data = companies.copy()
            data_daily_pct_change = data.pct_change().values[0:252]
            c = data_daily_pct_change.reshape(-1,1)
            while c.shape[0] < 252:
                c = np.vstack((c,c[-1]))
                if c.shape[0] == 252:
                    break
            cov_data.append(c)
    combined_matrix = np.hstack(cov_data)
    df = pd.DataFrame(combined_matrix).cov()

    return df
```

Appendix F

```python
def get_esg(investments):
    esg_data = []
    for i in investments:
        esg_score = yesg.get_historic_esg(i)
        if i == 'EQ_GIC':
            esg_data.append(min_esg)
        elif i == 'QuT_GIC':
            esg_data.append(min_esg)
        elif i in ['RBC_GIC', 'BMO_GIC', 'TD_GIC', 'CIBC_GIC']:
            esg_data.append(min_esg)
        elif esg_score is None:
            esg_data.append(min_esg)
        else:
            latest_esg = esg_score.iloc[-1]['Total-Score']
            esg_data.append(latest_esg)
    return esg_data
```

Appendix G

```
for m in final_return:
n = len(m)
print(m)
x = cp.Variable(n)


objective = cp.Maximize(m @ x )
constraints = [cp.sum(x) == 1,
            cp.quad_form(x, Sigma) <= R,
            esg @ x >= min_esg,
            x >= 0]

problem = cp.Problem(objective, constraints)

problem.solve(verbose=True, max_iters=1000000)

optimal_value = problem.value
optimal_solution = x.value
if optimal_value >= 0:
    print("Optimal return for broker:", np.round(optimal_value,3))
    print("Optimal solution:", np.round(optimal_solution,3))
    returns.append(np.round(optimal_value, 3))
    weights.append(np.round(optimal_solution, 3))
else:
    optimal_value = 0
    optimal_solution = np.zeros((1,n))
    print("Optimal return for broker:", np.round(optimal_value,3))
    print("Optimal solution:", np.round(optimal_solution,3))
    returns.append(np.round(optimal_value, 3))
    weights.append(np.round(optimal_solution, 3))
```