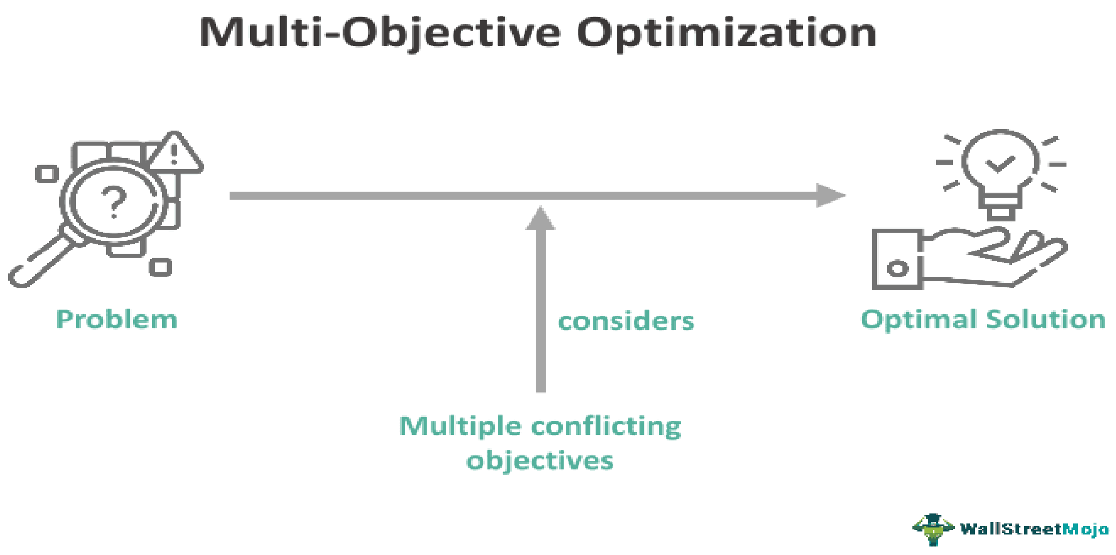
MULTI – OBJECTIVE PORTFOLIO OPTIMISATION

“Juggling investments like a circus performer: aiming for the trifecta of profit, growth, and avoiding pies in the face."



Introduction

Imagine you're a runner about to run a marathon. You have a few goals in mind for this marathon:

1. You want to finish the race as quickly as possible (that's like trying to make lots of money when you invest).
2.You also want to make sure you don't get too tired or hurt during the race (that's like not wanting to take too much risk).

3.But you have some special things you care about. You want to run in an eco-friendly way, so you don't harm the environment (like not littering). You also want to be a good friend and help other runners if they need it (like sharing your water with them).

Here's the challenge: You can't just focus on finishing the race ,forgetting about other factors. You have to find the best way to run that not only makes you finish quickly,but also stay safe, be eco-friendly, and be a good friend to others all at the same time.

So, you run at a good pace, not too fast to exhaust yourself but not too slow either. You might carry a reusable water bottle to be eco-friendly and help others by sharing your water when they need it.

Multi-objective portfolio optimization(MOPO) uses the same concept. It's about finding the best way to invest your money in different things (like stocks, bonds, or even saving for the future) so that you can achieve multiple goals at once, just like you want to run the marathon both quickly and responsibly. It's about finding a balance that leaves you happy,proud and satisfied and at the same time achieving your goal.

Multi-objective portfolio optimization is a complex and challenging problem in the field of finance and optimization. It involves selecting a combination of assets or investments that maximize multiple conflicting objectives simultaneously. These objectives often include maximizing returns, minimizing risk, and achieving other financial goals. The main idea is to find a balance between different objectives, as improving one objective may lead to a trade-off with another.

In traditional single-objective portfolio optimization, the goal is to either maximize returns for a given level of risk or minimize risk for a given level of returns. In multi-objective optimization, there can be more than two objectives, including objectives related to returns, risk, liquidity, and other financial metrics. Constraints may include limitations on asset allocation, sector exposure, or any other regulatory or investor-specific requirements.

History

Multi-Objective Portfolio Optimization can be pictured as a financial adventure through time. It started around the 1950s with a financial wizard named Harry Markowitz. He had this wild thought to find the perfect mix of investments which would give you most money while keeping minimal risk of losing it.

Then in the '80s and '90s, along came computers that made things way easier. People started using computer programs to figure out how to invest their money in the smartest way. It’s as if you have a super-smart assistant figuring everything out for you.
Moreover these days it wasn’t only about making money but there are other things like caring about the planet (called ESG investing) or paying less tax. So, now, it’s like trying to bake that same cake, but you also want it to be healthy (like being eco-friendly) and not too expensive (like saving taxes).

Moving on to the present, it’s not just computers helping us but there are super-smart machines that can process billions of pieces of information to help you make the best investment choices (called machine learning). It’s like having a superhero sidekick in the world of finance, making sure your money works hard for you.

Implementation

We looked into the history and definition.Now lets learn the implementation of Multi-Objective Portfolio Optimisation.

Step 1: Define your Objectives and their Constraints

- Start by listing your investment objectives. For example, we want to maximize returns and minimize risk here,but you can customize the objective as per your requirements.
- Specify needed constraints such as the total investment amount, asset allocation limits, and target minimum return.

Step 2: Collection of Data

- Gather historical data on the assets you want to include in your portfolio. This data should include all the past returns and risks (standard deviations) for each individual asset.

Step 3: Calculate Expected Returns and Risks

- Calculate the expected return and risk for each asset based on the historical data. The expected return is typically the average historical return, and risk is the standard deviation of returns.

Step 4: Create a Portfolio

- Create a table with different random portfolio combinations of assets. Vary the allocation percentages(weight) for each asset to cover a wide range of possibilities.

- Calculate the portfolio return and risk for each combination using weighted averages.

Step 5: Efficient Frontier

- Create a scatter plot with portfolio risk (x-axis) and return (y-axis).
- Plot each portfolio combination as a point on the graph.
- Use the best fitting line or curve to connect the points that represent the best trade-offs between risk and return. This curve is known as the **Efficient Frontier*.**

** The efficient frontier is a concept in finance that represents the set of optimal portfolios offering the highest expected return for a given level of risk, or the lowest risk for a given level of expected return. It illustrates the trade-off between risk and reward in portfolio optimization.*

Step 6: Optimization

- Implement mathematical optimization algorithms (e.g., quadratic ,linear programming) to find the portfolios on the Efficient Frontier that best aligns with your objectives and constraints.
- These portfolios are known as **Pareto-optimal** portfolios.

Step 7: Plot Pareto-optimal Portfolios

- On the Efficient Frontier graph, plot the Pareto-optimal portfolios obtained through optimization. These are the portfolios that represent the best return to risk ratio.

Step 8: Choose a Portfolio

- Depending on your risk tolerance and return requirements, select one of the Pareto-optimal portfolios from the graph.
- This chosen portfolio will be the optimal allocation of assets that aligns with your required objectives and constraints.

We've understood the crux and the process,now let's move onto the mathematics behind it.

Mathematical optimisation

The mathematical formulation of multi-objective portfolio optimization involves defining objective functions, decision variables, and constraints. Below is a simplified representation of the mathematical model for multi-objective portfolio optimization:

Objective Functions:

For the mathematical part, consider k different objective functions $f_1(x), f_2(x), \ldots, f_k(x)$ where x is the vector of decision variables representing the portfolio weights.
Typically, the objectives include maximizing returns and minimizing risk, but they can also include other financial or non-financial goals. These objectives are:

- $\boldsymbol{f_1(x)}$: the return objective, e.g., maximizing expected portfolio return.

- ***$f_2(x)$*** : the risk objective, e.g., minimizing portfolio risk (variance or standard deviation).
- ***$f_3(x)$*** : represents additional objectives such as achieving a target level of risk-adjusted return, sector exposure constraints, or other specific criteria.

Decision Variables:
Let x be a vector of decision variables representing the portfolio weights assigned to each asset in the portfolio. The sum of these weights should equal 1 to ensure that the portfolio is fully invested.

$$\mathrm{sum}(x_1, x_2, \ldots\ldots, x_n) = 1$$

Where x_i is the weight of i-*th* asset in the portfolio and n is the total number of assets.

Constraints:
- The sum of all weights should always be equal to 1.
- Minimum and maximum investment limits for all assets their weights should be $x^{min} \leq x \leq x^{max}$
- Other constraints like constraints on sector exposure, asset class exposure, or other specific requirements.

Solving the problem:
To solve the multi-objective problem, you need to find a set of portfolio weights (x) that optimizes the multiple objective functions simultaneously. This often involves trade-offs between the objectives, and different mathematical techniques can be used to address this trade-off, such as scalarization methods.

A common approach is to use scalarization to convert the multi-objective problem into a single-objective problem by using weighted combinations of the objectives. For example, you can create a weighted sum of the objectives:

$$Z = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \ldots\ldots + w_k \cdot f_k(x)$$

We then maximize the function Z by adjusting the weights $w_1, w_2, .., w_k$
Here, $w_1, w_2, .., w_k$ are weights representing the **importance** of each objective.
Adjusting these weights allows you to explore the trade-offs between objectives and find different solutions on the **Pareto front**.

The **Pareto front represents a set of solutions where one objective cannot be improved without degrading another. It illustrates trade-offs between conflicting objectives.*

Example

For ease of understanding,let's take two objectives instead of more.
Next,we continue with the following steps:
Step 1: Define the Objective Function:
The objective function is a mathematical expression that quantifies what you want to maximize or minimize. It typically involves decision variables.

Our Objective Function:
Maximize: $Z = 2x_1 + 3x_2$

Step 2: Identify Decision Variables:
Decision variables are the parameters you want to optimize. They are often denoted by symbols (e.g., x, y) and can represent quantities, allocations, or actions.In our example ,they are **x_1 and x_2.**

Step 3: Specify Constraints:
Constraints are conditions or limitations that must be satisfied. They can be equations or inequalities involving the decision variables.
Here the constraints are:

$$2x_1 + x_2 \leq 6$$
$$x_1 + 2x_2 \leq 8$$

Step 4: Formulate the Optimization Problem:

Combine the objective function and constraints to create a comprehensive mathematical expression.
Our overall question:

Maximize: $Z = 2x_1 + 3x_2$

Subject to:

$2x_1 + x_2 \leq 6$

$x_1 + 2x_2 \leq 8$

Step 5: Choose the Optimization Technique:

Select the appropriate optimization method for solving the problem. Common techniques include linear programming (LP), quadratic programming (QP), mixed-integer linear programming (MILP), nonlinear programming (NLP), and more.

Step 6. Solve the Optimization Problem:

Apply the chosen optimization technique to find the optimal solution. Software packages like MATLAB, Python (using libraries like SciPy), or specialized optimization solvers can be used.

Step 7: Evaluate the Solution:

After solving the optimization problem, you will obtain values for the decision variables. Evaluate the objective function to determine the optimal value.

In our Example:

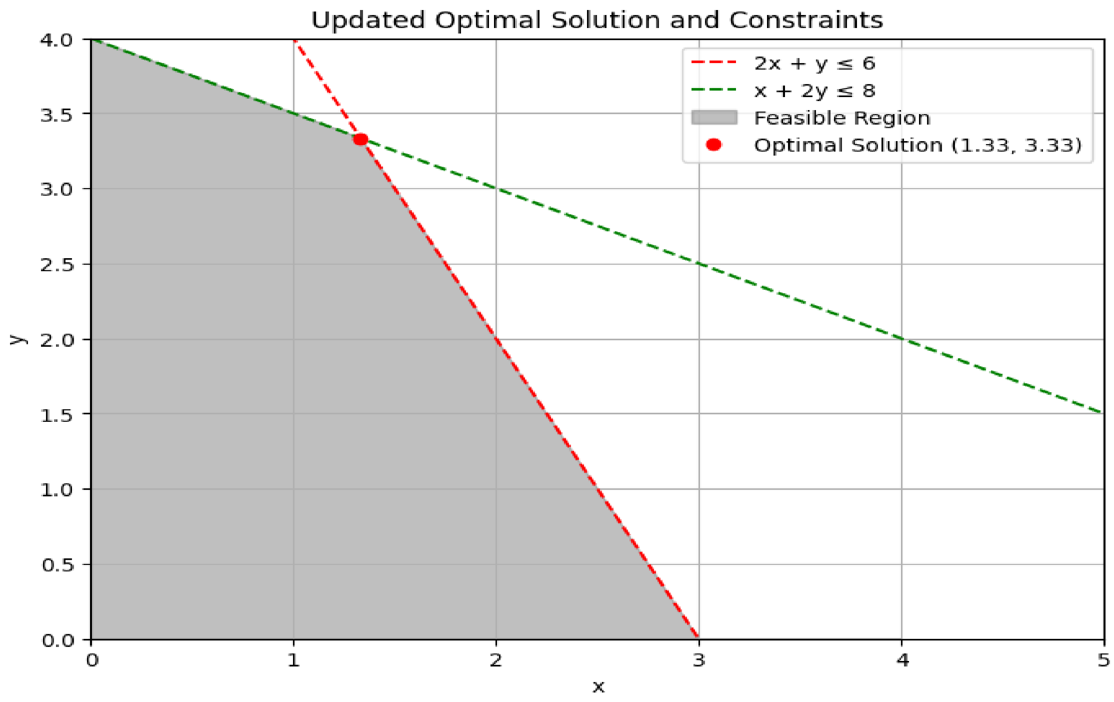
Optimal Solution (x_i*):

$x_1^* = 1.33$

$x_2^* = 3.33$

Maximized Value: $Z^* = 2x_1^* + 3x_2^* = 2(1.33) + 3(3.33) = 12.67$

Plotting our results ,we get:



Run the full mathematical model here: Model

Step 8: Interpret the Results:
Understand the implications of the optimal solution in the context of your problem. Does it make sense given your objectives and constraints?

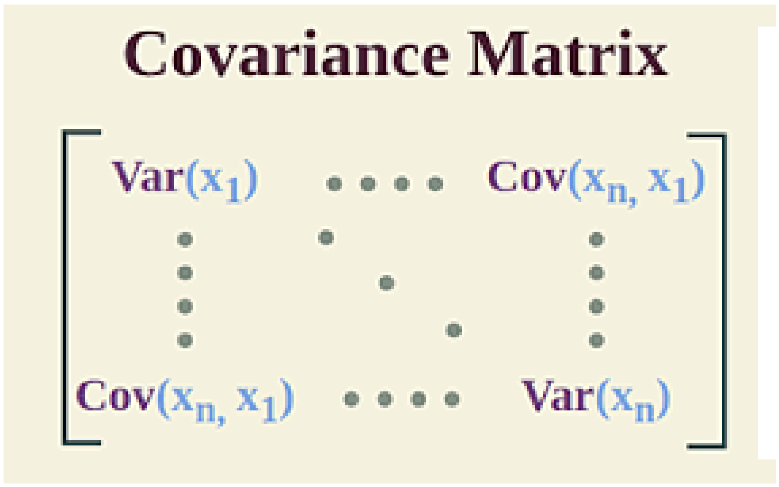
The above steps provide a simplified overview of the process. In practice, optimization problems can be much more complex, involving nonlinear objective functions, multiple constraints, and discrete variables. The specific formulae and methods used will depend on the nature of the problem and the chosen optimization technique.

Code Implementation of MOPO:

<u>Overview of Code</u>

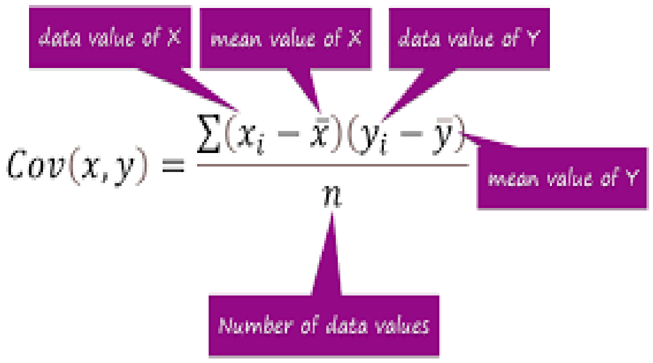
Before jumping to the full code implementation let us look at an overview of what we are trying to do with the code and then we will move on to the actual code.

1. We start by importing the necessary libraries such as numpy, matplotlib and cvxpy. Cvxpy is used to formulate and solve convex optimization problems.
2. Then we decide on how many assets(***'n_assets'***) we want to consider and then save their expected returns (***'expected_returns'***) in an array. We saved the covariance among the assets in a matrix (***'cov_matrix'***)



Covariance Matrix

$$\begin{bmatrix} Var(x_1) & \cdots & Cov(x_n, x_1) \\ \vdots & \ddots & \vdots \\ Cov(x_n, x_1) & \cdots & Var(x_n) \end{bmatrix}$$



$$Cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n}$$

3. We then defined the maximum allowed portfolio risk (variance) which is ***'risk_tolerance'*** and the desired level of expected return from the portfolio '***target_return'***.
4. The weights of the assets in the portfolio are represented by '***weights'*** and it is declared as a cvxpy.Variable.
5. Then we define ***'expected_return'*** which is calculated as the sum of expected returns weighted by the portfolio weights and '***portfolio_risk'*** is defined using the quadratic form to calculate portfolio risk (variance).

6. Then we set the constraints for the model to train on. The constraints are
    - The sum of all weights should be 1 (fully invested portfolio)
    - All the weights should be greater than equal to 0 (only long positions)
    - Expected return should be more than or equal to target return
    - Portfolio risk should be less than equal to risk tolerance.
7. Defined the objective which is to maximize the expected return.
8. Create this problem using the ***'cvxpy.Problem'*** and the arguments as ***'objective'*** and ***'constraints'.***
9. Solve the multi-objective optimization problem using '***problem.solve()'***. This finds the portfolio weights that maximize expected return while satisfying constraints.Extract and print the portfolio weights, expected return, and risk (standard deviation) of the portfolio.
10. Extract and print the portfolio weights, expected return, and risk (standard deviation) of the portfolio.
11. Calculate and plot the efficient frontier:

Detailed code:

- **Importing libraries,declaration of assets and input of expected returns as well as covariance matrix.**

```
#importing necessary libraries
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt

# Number of assets
n_assets = 5

# Define expected returns and covariance matrix (replace with your data)
```

```
expected_returns = np.array([0.08, 0.10, 0.09, 0.09, 0.11])
cov_matrix = np.array([[0.04, 0.02, 0.015, 0.01, 0.025],
                       [0.02, 0.03, 0.01, 0.015, 0.02],
                       [0.015, 0.01, 0.02, 0.015, 0.01],
                       [0.01, 0.015, 0.015, 0.03, 0.02],
                       [0.025, 0.02, 0.01, 0.02, 0.04]])
```

- **Defining returns,risk,variable weights, as well as the optimisation problem(setting total sum of weights=1)**

```
# Define the risk tolerance and target return
risk_tolerance = 0.06
target_return = 0.05

# Define the variable for portfolio weights
weights = cp.Variable(n_assets)

# Define the expected return and risk objectives
expected_return = cp.sum(expected_returns @ weights)
portfolio_risk = cp.quad_form(weights, cov_matrix)

# Define the optimization problem
constraints = [cp.sum(weights) == 1, expected_return >= target_return,
portfolio_risk <= risk_tolerance,weights >= 0]
objective = cp.Maximize(expected_return)
problem = cp.Problem(objective, constraints)
```

- **Solving the problems and printing the optimized weight distribution**

```
# Solve the multi-objective optimization problem
problem.solve()

# Extract portfolio weights
portfolio_weights = weights.value

# Print the results
```

```
print("Portfolio Weights:")
print(portfolio_weights)
print("Portfolio Expected Return:", expected_return.value)
print("Portfolio Risk (Standard Deviation):",
np.sqrt(portfolio_risk.value))
```

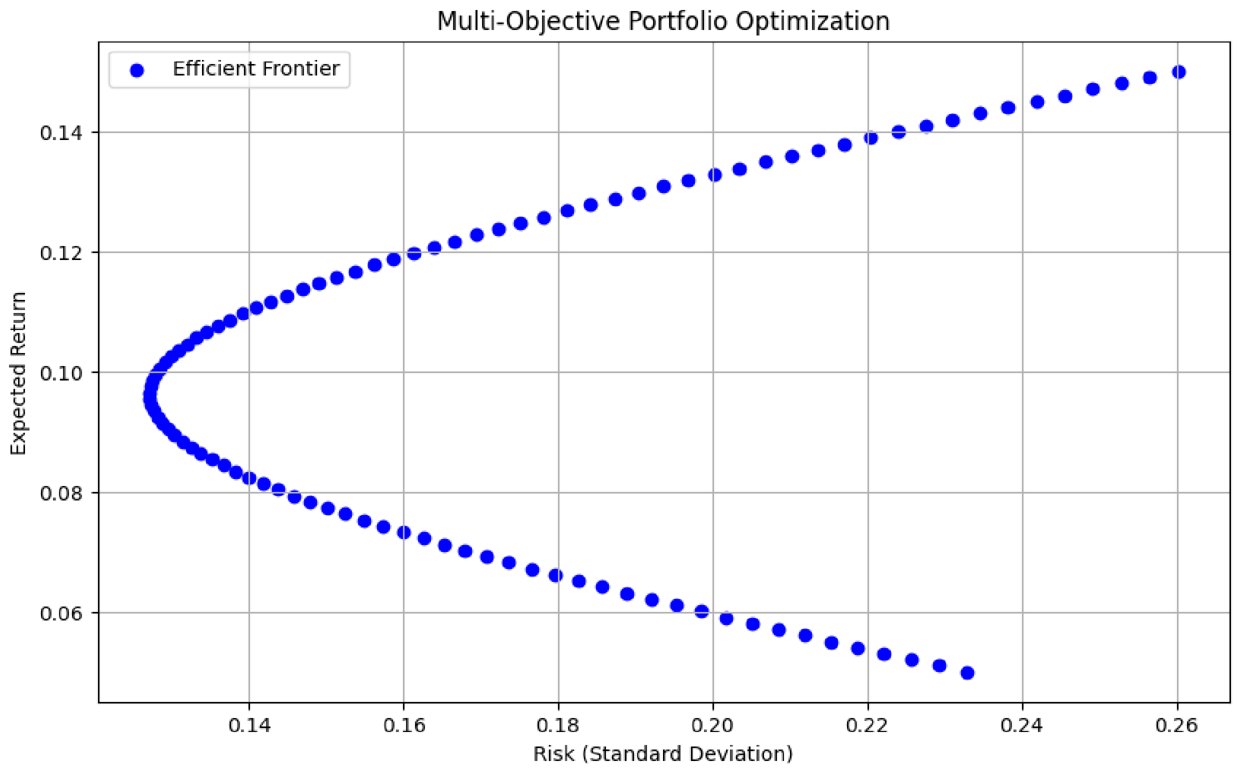
- **Plotting the efficient frontier(max risk and return combination)**

```
# Plot the efficient frontier
risk_values = []
return_values = []

for r in np.linspace(0.05, 0.15, 100):
    expected_return = cp.sum(expected_returns @ weights)
    portfolio_risk = cp.quad_form(weights, cov_matrix)
    constraints = [cp.sum(weights) == 1, expected_return == r]
    problem = cp.Problem(cp.Minimize(portfolio_risk), constraints)
    problem.solve()
    risk_values.append(np.sqrt(portfolio_risk.value))
    return_values.append(r)

plt.figure(figsize=(10, 6))
plt.scatter(risk_values, return_values, c='b', marker='o',
label='Efficient Frontier')
plt.xlabel('Risk (Standard Deviation)')
plt.ylabel('Expected Return')
plt.title('Multi-Objective Portfolio Optimization')
plt.grid()
plt.legend()
plt.show()
```

- **Finally,presenting our hard work!**



Portfolio Weights:
[3.96480619e-09 2.57520588e-09 4.42806578e-09 5.17110470e-09
9.99999984e-01]
Portfolio Expected Return:
0.10999999663324
Portfolio Risk (Standard Deviation):
0.19999999826380524

Run the full code here : CODE

Conclusion and Insights

- Unlike single objective portfolio optimisation ,there is no single best solution of MOPO,**but a range of solutions with different risks and returns.**
- It doesn’t directly optimize risk and return but takes in a **variety of utility functions** in order to cater to the preferences of the investor.

- It can consider goals not only associated with financial metrics,but also **specific wealth,retirement readiness, or philanthropic targets.**
- Investors can try and test **different economic markets** to see how the portfolios perform.
- Investors can explore efficient frontier for **multiple objectives** simultaneously,often having conflicting goals.
- It also has a **dynamic setting** which helps it to keep itself updated with the trends.
- MOPO considers **trading costs as well and liquidity constraints** which single objective portfolio optimisation doesn't.

While MOPO in itself,is a commendable and powerful tool ,it also has some downsides.

- ➔ **Complexity-** Due to its multi objective ability,this requires specialized tools and algorithms to implement.
- ➔ **Ambiguity -** MOPO results in a range of solutions with various **parieto-optimal solutions,**selecting the best solution can be difficult.
- ➔ **Data accuracy-** The input data (returns,volatility,correlations) are very sensitive and significantly impact the optimization results.
- ➔ **Runtime -** Solving MOPO problems can be computationally intensive,especially for large portfolio and complex objectives.

All in all ,Multi-Objective Portfolio Optimisation redefines investment decisions by harmonizing risk,return and diverse objectives.Through Pareto-Efficient Solutions it helps the portfolio to have a balanced trade off,thus helping him/her to make personalized,optimum and informed strategies.

Follow us to read more such interesting blogs:

Quant Club Instagram
Quant Club Facebook
Quant Club Linkedin

References

- **https://medium.com/**
- **https://www.mdpi.com/2297-8747/26/2/36**
- **https://www.sciencedirect.com/science/article/pii/S2210650222001110#:~:text=A%20multi%2Dobjective%20optimization%20model,to%20maintain%20effective%20continuous%20investment.**
- **https://www.hindawi.com/journals/mpe/2017/4197914/**
- **https://colab.research.google.com/?utm_source=scs-index**

THANK YOU!