Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/TQE.2020.DOI

Backtesting quantum computing algorithms for portfolio optimization

GINÉS CARRASCAL [1, 2], PAULA HERNAMPEREZ [3], GUILLERMO BOTELLA[4], (Senior Member, IEEE), and ALBERTO DEL BARRIO.[4],(Senior Member, IEEE)

[1]IBM Quantum, IBM Consulting España, 28830 Madrid, SPAIN
[2]Department of Informatic Systems and Computation, Faculty of Informatics, Complutense University of Madrid, 28040 Madrid, SPAIN
[3]CARTIF, 47151 Valladolid, SPAIN
[4]Department of Computer Architecture and Automation, Faculty of Informatics, Complutense University of Madrid, 28040 Madrid, SPAIN

Corresponding author: Ginés Carrascal (email: gines_carrascal@es.ibm.com).

ABSTRACT The development of quantum computers represents a breakthrough in the evolution of computing. Their graceful processing capacity will help to solve some problems impossible until now because the algorithms that calculate their solution require too much amount of memory or processing time. In portfolio theory, the investment portfolio optimization problem is one of those problems whose complexity grows exponentially with the number of assets. In this work we analyze the Variational Quantum Eigensolver algorithm, applied to solve the portfolio optimization problem, running on simulators and real quantum computers from IBM. We compare the results with three other classical algorithms for the same problem, running one equivalent condition. By backtesting classical and quantum computing algorithms, we can get a sense of how these algorithms might perform in the real world. This work explores the backtesting of quantum and classical computing algorithms for portfolio optimization and compares the results. The benefits and drawbacks of backtesting are discussed, as well as some of the challenges involved in using real quantum computers of more than 100 qubits. Results show quantum algorithms can be competitive with classical ones, with the advantage of being able to handle a large number of assets in a reasonable time on a future larger quantum computer.

INDEX TERMS Backtesting, CVaR VQE, Portfolio Optimization, Quantum Computing, VQE

I. INTRODUCTION

THROUGHOUT history, humans have used tools to solve complex problems. As the complexity of problems has increased, so has the processing capacity of computers. However, there are still many problems that current computers cannot solve. Gordon Moore's prediction that the number of transistors in a microprocessor would double every two years has held true and has helped increase the processing capacity of computers. However, this trend is expected to end in the near future due to limits on the size of transistors [1].

The new generation of computers, quantum computers, still under development, promises a drastic advance in computing development as quantum computing has the potential to drastically decrease computational cost and execution time [2]. For this reason, the study of this new generation of computers is currently of great interest.

One of these complex problems is portfolio optimization. It is the process of selecting the best mix of investments to maximize return and minimize risk. This can be done by using a variety of techniques, including mathematical modeling, statistical analysis, and machine learning.

One approach to solving complex financial services problems is to use quantum computing algorithms [3]. These algorithms can help identify patterns and relationships that would be difficult to find with traditional methods [4]. By backtesting these algorithms, it is possible to see how they would have performed in different market conditions. This information can then be used to make better investment decisions in the future.

Traditional portfolio optimization methods are based on classical algorithms that are not well suited to handle the large amounts of data involved in modern financial portfolios [5]. Quantum computers, with their ability to perform massively parallel computations, offer the promise of more efficient and effective portfolio optimization.

The Variational Quantum Eigensolver (VQE) is one quantum algorithm that has been proposed for use in portfolio optimization. VQE is a hybrid quantum-classical algorithm

that uses a quantum computer to find the lowest energy state of a system, which in this case would be the optimal portfolio [6]. To test whether VQE can outperform classical algorithms for optimization, researchers at IBM conducted a series of experiments on IBM Quantum devices [7]. They found that VQE was able to find the global minimum in most cases and theoretically can outperform classical algorithms when the number of assets in the portfolio was large. While further research is needed to determine if VQE can be successfully applied to real-world financial portfolios, these results suggest that quantum computers may one day play an important role in helping us make better investment decisions.

Backtesting is crucial in evaluating the performance of a portfolio optimization algorithm. It involves using historical data or Monte Carlo simulations to determine the algorithm's ability to find the optimal portfolio and assess its robustness. When backtesting, one should consider using a large enough dataset, paying attention to results from different tests, and including a risk analysis to understand potential downsides. However, past performance does not guarantee future results.

This work explores the backtesting of quantum and classical computing algorithms for portfolio optimization and compares the results. We address the challenges involved in using current real quantum computers of 27 and 127 qubits, especially the definition of an ansatz with low length and the search for a suitable path in the real chip to map the circuit, in terms of quality and connectivity, to avoid a bigger circuit in the transpilation phase. Results show quantum algorithms can be competitive with classical ones, with the advantage of being able to handle a large number of assets in a reasonable time on a future bigger quantum computer.

This paper is organized as follows: Section II introduces the idea of portfolio optimization. Section III explains the implications of quadratic programming. Section IV describes the possibilities of using quantum computing for portfolio optimization. Section V introduces backtesting as well as its benefits and drawbacks. Sections VI lists the various traditional methods that are used to solve this problem, and Section VII describes the quantum algorithms that can be an alternative. Section VIII describes our methodology based on backtesting comparison of different classical and quantum algorithms for portfolio optimization. Section IX details the implementation of the different algorithms to be comparable, and the considerations we take with real quantum computers. Section X presents the results. Section XI contains our conclusions.

II. PORTFOLIO OPTIMIZATION

Portfolio optimization is the process of selecting the best mix of investments. To identify assets with low correlations and determine the optimal allocation of assets to maximize return on investment while minimizing risk. This selection must be made according to the client's characteristics and risk aversion [8].

The goal of portfolio optimization is to find the optimal balance between risk and return. This can be a difficult task, as there are many factors that need to be considered, such as investment objectives, constraints, market conditions, and individual preferences. This typically involves analyzing the potential risks and returns of different assets, such as stocks, bonds, and real estate, and then creating a balanced portfolio that aligns with the investor's risk tolerance and financial goals.

The process of portfolio optimization involves several key steps. First, the investor must define their investment objectives, including their desired level of risk and return. This will help to determine the overall mix of assets that should be included in the portfolio. Next, the investor must identify the various assets that are available for inclusion in the portfolio and analyze their potential risks and returns. Then, the investor must determine the optimal mix of assets that will form the portfolio. This typically involves using mathematical optimization techniques to find the combination of assets that are expected to provide the best return for a given level of risk. Finally, the investor must implement the chosen portfolio and monitor it regularly to ensure that it is still meeting the investment objectives. This may involve adjusting the portfolio, such as buying or selling assets, in order to maintain the desired level of risk and return.

While mathematical optimization can be a powerful tool for portfolio optimization, there are several limitations to be aware of. One major limitation is that optimization models are based on assumptions and historical data, and they may not accurately predict future market conditions [9]. In other words, the optimization process can only provide a best guess as to how a portfolio is likely to perform, and there is no guarantee that it will actually provide the expected return. Another limitation is that those optimization models are typically based on a static view of the market, which means that they do not consider any changes that may occur over time [10]. This can make it difficult for investors to adapt to changing market conditions and to respond to unexpected events.

III. QUADRATIC PROGRAMMING

Quadratic programming is a mathematical optimization technique that is often used in portfolio optimization. It is a type of nonlinear optimization that involves minimizing or maximizing an objective function subject to constraints. It is an NP problem [11].

In the context of portfolio optimization, the objective function is typically a measure of portfolio performance, such as expected return or risk-adjusted return. The constraints typically include limits on the amount of each asset that can be included in the portfolio, as well as other factors such as the overall level of risk or the level of exposure to specific assets or sectors.

To solve a quadratic programming problem, an optimization algorithm is used to find the values of the variables (in this case, the amounts of each asset in the portfolio) that minimize or maximize the objective function, subject to the constraints. The resulting portfolio is then expected

to provide the best possible performance, given the specified constraints. Quadratic programming has been widely used in portfolio optimization because it can handle a large number of variables and constraints, and it can provide solutions that are efficient and easy to interpret. It is also relatively fast and scalable, which makes it well-suited to the demands of modern investment management [12]. However, it has several limitations. It only works for differentiable objective functions and constraints. Besides, it assumes that they are convex (i.e., have only one global minimum or maximum). This means that it may not be suitable for solving problems with discontinuous or non-differentiable functions, or non-convex problems. It is important for users to be aware of these limitations and use quadratic programming appropriately.

The number of assets that can be included in portfolio optimization using quadratic programming is limited by the computational power of the computer and the complexity of the problem. The exact number of assets that can be considered will depend on the specific implementation and available computing resources, but some quadratic programming solvers can handle problems with thousands of variables [13].

Quadratic programming algorithms may get stuck at a local minimum or maximum, rather than finding the global optimum. A local minimum (or maximum) is a point in the search space where the objective function has a lower (or higher) value than in the immediate surrounding area, but is not the global minimum (or maximum) of the function. This limitation can arise due to the specific optimization algorithm being used, as well as the nature of the objective function and the constraints. To try to avoid getting stuck at local optima, quadratic programming algorithms may use heuristic techniques such as random restarts or simulated annealing, but these techniques do not guarantee finding the global optimum. It is important for users to be aware of this potential limitation and use appropriate algorithms and heuristics to try to avoid it.

IV. QUANTUM COMPUTING FOR PORTFOLIO OPTIMIZATION

Quantum computing is a new field that uses the principles of quantum mechanics to perform calculations that are beyond the reach of classical computers. It has the potential to solve complex problems, including some problems in finance and investment management. In the context of portfolio optimization, quantum computing could be used to solve large or complex optimization problems, allowing for more accurate and sophisticated portfolio models and more informed asset allocation and risk management decisions. It could also be used to develop new optimization algorithms and techniques that take advantage of the unique properties of quantum systems, potentially providing more accurate and efficient solutions to portfolio optimization problems. While quantum computing is still in the early stages of development, it has the potential to significantly improve portfolio optimization.

The Variational Quantum Eigensolver (VQE) is an algorithm for quantum computing that is often used to solve optimization problems. It is based on the principle of quantum mechanics, which allows for the representation of complex mathematical functions as quantum states [14].

The VQE algorithm works by representing the objective function and the constraints of the optimization problem as parameterized quantum states, and then using a classical optimizer to find the values of the variables (in this case, the amounts of each asset in the portfolio) that minimize or maximize the objective function, subject to the constraints.

A quadratic program (QP) is a type of mathematical optimization problem that involves minimizing or maximizing a quadratic objective function subject to linear constraints. A quadratic binary optimization (QUBO) problem is a similar type of optimization problem, but it involves binary variables (i.e., variables that can take on only two values, such as 0 or 1) and quadratic objectives without constraint functions.

Our problem is: VQE can solve a QUBO, not a QP.

In some cases, it is possible to transform a quadratic program into a quadratic binary optimization (QUBO) problem by replacing the continuous variables in the QP with binary variables. This can be done by defining a set of binary variables that represent the possible values of the continuous variables, and then using these binary variables in the objective and constraint functions of the QP.

For example, suppose we have a quadratic program with a continuous variable x that can take on any real value. We can define binary variables, $x_0...x_N$, that represent the possible values of x (i.e., $x_0...x_N$ can be the binary representation of x real value with certain precision). We can then rewrite the objective and constraint functions of the QP using the binary variables $x_0...x_N$, to create a QUBO problem.

As a general rule, the process of transforming a quadratic program into a QUBO problem can be complex, but it can provide valuable insights and more efficient solutions to the optimization problem. By leveraging the unique properties of binary variables and quadratic functions, QUBO optimization can provide more accurate and efficient solutions to a wide range of optimization problems, including those in finance and investment management.

V. BACKTESTING: AN ESSENTIAL TOOL FOR VALIDATION

Backtesting is a method of evaluating the performance of a trading strategy or investment portfolio by simulating its performance on historical data. It allows traders and investors to see how the strategy or portfolio would have performed under different market conditions and can be used to validate the effectiveness of the strategy or portfolio and to identify any potential weaknesses. To conduct a backtest, a trader or investor specifies the parameters of the strategy or portfolio and simulates its performance over a specific time period using historical data. Many authors use backtesting to show the benefits of their proposed algorithms or strategies [15], [16].

While backtesting can be a useful tool for evaluating and optimizing strategies or portfolios, it is important to note

that past performance is not necessarily indicative of future results and that backtesting can be subject to biases that can affect the accuracy of the results.

Survivorship bias is a type of bias that occurs when a sample of data used in a backtest only includes assets that have survived until the present time and excludes assets that have been removed from the market or are no longer being traded. This can lead to an overestimation of the performance of the trading strategy or investment portfolio being tested, as it does not take into account the performance of assets that did not survive. To avoid survivorship bias, it is important to use a representative sample of data that includes both surviving and non-surviving assets when conducting a backtest. This can help to provide a more accurate representation of the performance of the strategy or portfolio being tested [17].

Look-ahead bias, also known as peeking or snooping bias, is a type of bias that occurs when a trading strategy or investment portfolio is evaluated using data that was not available at the time the decisions were made. In the context of backtesting, look-ahead bias can occur when the backtest includes data that were not available to the trader or investor at the time the trades or investments were made. This can lead again to an overestimation of the performance of the strategy or portfolio being tested, as it does not accurately reflect the information and resources that were available to the trader or investor at the time the decisions were made. To avoid look-ahead bias, it is important to use only data that was available at the time the trades or investments were made when conducting a backtest. This can help to provide a more accurate representation of the performance of the strategy or portfolio being tested [18].

As a result, backtesting can be a useful tool for evaluating and optimizing trading strategies and investment portfolios, but it is important to keep in mind its limitations and to use it in conjunction with other tools and analyses. Although there is a work from Owhadi-Kareshk and Boulanger [19] describing a tool for backtesting classical and quantum annealing algorithms for portfolio optimization, they make an example of use for their software and not an actual analysis of the algorithms. To the best of our knowledge, this is the first work that performs a systematic backtesting comparison of classical and quantum portfolio optimization algorithms.

VI. CLASSICAL ALGORITHMS FOR PORTFOLIO OPTIMIZATION

Next, we will explain three classical algorithms used to solve the portfolio problem.

A. MOVING AVERAGE STRATEGY

A moving average [20] is a technical analysis tool that is used to smooth out price data by calculating the average price over a specified period of time. It is often used to help identify trends, spot trend reversals, and smooth out price fluctuations.

The concept of moving averages or Moving Average (MA) refers to the calculation of the average in a time window of a time series. That is, for a time series, such as the closing prices of an asset, which we will denote x_t on each day t, the MA with a time window τ, is calculated as

$$\mathrm{MA}_t(X) = \frac{1}{\tau} \sum_{s=t-\tau+1}^{s=t} x_s.$$

For example, a 50-day SMA would be calculated by adding up the closing prices for the past 50 days and then dividing the sum by 50.

For the implementation of the Simple Moving Average Strategy (SMAS) algorithm, the MA of each of the assets considered at the present moment is calculated. We will randomly choose the agreed number (budget) of assets from among those that are above their MA. These assets will be the ones that make up our investment portfolio, which we consider to have the same weight in the portfolio.

The SMAS algorithm is widely used in the financial world due to its easy understanding [21]. Moving averages can be used in various ways in technical analysis, such as identifying trends, spotting trend reversals, and identifying support and resistance levels. However, it is important to keep in mind that moving averages are a lagging indicator, meaning that they are based on past price data and may not always accurately predict future price movements.

B. RISK-RETURN OPTIMIZATION (MARKOWITZ MODEL)

The main theory on which portfolio optimization models are based is known as modern portfolio theory. This new approach was introduced by Markowitz who suggested analyzing both profitability and risk as a whole for the first time, which caused a revolution in the financial world, giving rise to the new theory of portfolio optimization [22], [23].

This type of algorithm optimizes the risk-return ratio (MVO, Mean-Variance optimization). Risk should be minimized and profitability should be maximized at the same time.

Profitability is defined as a financial ratio that measures the ability to generate profits, the measure of profitability will be the expected return of portfolio prices. The criterion that Markowitz's theory uses to choose the assets that make up the portfolio is their performance in relation to others. These must have low correlation between them and be diversified. This asset diversification process consists of dividing the investment among different stocks to reduce their overall exposure to risk without reducing the expected return.

Mathematically, a Markowitz model can be defined as follows:

$$\begin{gathered} \max\ R_p - q\sigma_p^2 \\ s.t. \\ x_i \geq 0, i = 1, ..., n \\ \textstyle\sum_{i=1}^{n} x_i = 1, \end{gathered} \tag{1}$$

where q is a risk aversion factor, R_p is the total return and σ_p^2 is the variance of the portfolio. Besides, x_i are the amount

invested in each asset, subject to budget and non-negativity restrictions.

Prior to Markowitz's model, portfolio selection focused primarily on returns generated by investment opportunities. Markowitz maintains the emphasis on these, at the same time begins to consider risk as an important factor and the variance of a portfolio as a way of measuring risk.

Markowitz was the first to demonstrate how the variance of a portfolio can be reduced through the impact of diversification. It is proposed that investors focus on selecting portfolios based on their overall risk-reward rather than selecting portfolios that individually have attractive risk-return characteristics.

Markowitz's model accommodates the choice between risk and return of the investor. Regardless of the preference of the inverter, all these are under what is called the Markowitz efficient boundary. It is defined as the set of efficient portfolios, that is, those that offer a higher expected return according to the different levels of risk that can be assumed.

This relates volatility and profitability of a portfolio, that is, the benefits that the investor can obtain and the risks he must face to have it. This relationship must be positive, the higher the profitability, the larger the risk. At the border, risk-free assets are portfolios of minimum variance and are located to the left of the border (Fig. 1).

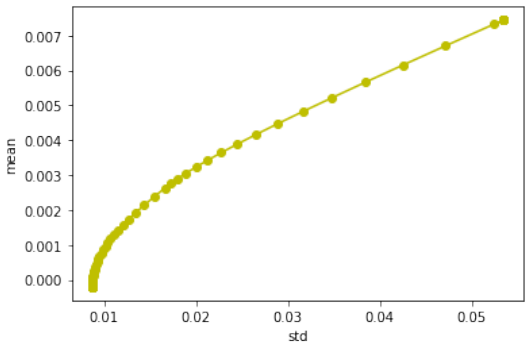

FIGURE 1. Graphical representation of the efficient frontier.

In the context of portfolio management, each investor must decide where to position. An investor with a conservative profile will opt for a maximum return and a minimum risk, however, someone riskier is positioned to the right of the border to obtain a higher return but with the possibility of suffering more losses.

C. SHARPE RATIO OPTIMIZATION

The Sharpe Ratio is a measure that allows the profitability of an investment portfolio to be related to its risk [24]. The Sharpe Ratio is calculated by dividing the portfolio's excess return (the difference between its return and the risk-free rate of return) by its standard deviation, which is a measure of its volatility. It is calculated as

$$\text{SR} = \frac{\mu - R_f}{\sigma},$$

where μ represents the return of the portfolio made up of assets with strictly positive risk, R_f is the return of the asset without risk and σ the volatility of the portfolio.

A higher Sharpe ratio indicates a higher risk-adjusted return, meaning that the portfolio has generated a higher return per unit of risk.

The Sharpe Ratio Optimization (SRO) algorithm is based on maximizing the Sharpe Ratio to improve the return-risk ratio of our investment. The resulting portfolio is known as the market portfolio (optimal risky portfolio) [25].

This can be achieved through a variety of methods, such as diversifying the portfolio across a range of assets, using risk management techniques, or selecting assets with lower volatility.

It is important to note that while maximizing the Sharpe ratio may be a useful goal in some cases, it is not always the best approach to portfolio optimization. For example, an investor who is seeking to maximize the expected return of their portfolio may be willing to accept a higher level of risk in order to achieve higher returns. In these cases, maximizing the Sharpe ratio may not be the most appropriate goal [26].

VII. QUANTUM ALGORITHMS FOR PORTFOLIO OPTIMIZATION

The routines presented here are hybrid quantum-classical and compatible with each other.

A. VARIATIONAL QUANTUM EIGENSOLVER (VQE)

The Variational Quantum Eigensolver (VQE) algorithm is an algorithm used to calculate the smallest eigenvalue of a Hamiltonian, that is, its ground state.

Some of the properties of these matrices are the following:

- The eigenvalues are always real.
- The eigenvectors associated with distinct eigenvalues are orthogonal.

If H is a Hamiltonian (i.e Hermitian matrix), and λ_i and $|\boldsymbol{\psi_i}\rangle$ with $i = 1, ..., n$ are their eigenvalues and corresponding unit eigenvectors, it is possible to represent the matrix as

$$H = \sum_{i=1}^{n} \lambda_i |\boldsymbol{\psi_i}\rangle\langle\boldsymbol{\psi_i}|. \tag{2}$$

The expected value of a matrix of H and a vector $|\boldsymbol{\phi}\rangle$ is defined as the value

$$\langle H\rangle_{\boldsymbol{\phi}} = \langle\boldsymbol{\phi}|H|\boldsymbol{\phi}\rangle = \sum_{i=1}^{n} \lambda_i |\langle\boldsymbol{\psi_i}|\boldsymbol{\phi}\rangle|^2. \tag{3}$$

It is evident, therefore, that for all $\boldsymbol{\phi}$, $\langle H\rangle_{\boldsymbol{\phi}} \geq \lambda_{min}$ where λ_{min} is the smallest of the eigenvalues of H, since $|\langle\boldsymbol{\psi_i}|\boldsymbol{\phi}\rangle|^2 \geq 0$. Therefore, $\langle H\rangle_{\phi}$ are upper bounds of λ_{min} and also, if the eigenvalue λ_{min} is not double in the matrix, there will exist a unique eigenvector $\boldsymbol{\psi_{min}}$ that verifies

$$\langle H\rangle_{\psi_{min}} = \lambda_{min}.$$

To find this eigenvector, the VQE algorithm uses what is known as a variational form or ansatz. It is a circuit that is

used to go iteratively from one estimate of ψ_{min} to another, seeking that in each assessment the expected value of the eigenvector is lower. There are different variational forms, but all depend on a series of parameters that grow with the number of qubits. Adjusting these parameters to the specific problem is done with a classical optimizer, hence the VQE is a hybrid algorithm.

Given a Hamiltonian H, of dimension $2^N \times 2^N$, which we can write in quantum form as a function of Pauli matrices $(\sigma_x, \sigma_y, \sigma_z)$ as in (4) where $i, j = 1, ..., N$; $\alpha, \beta \in \{x, y, z\}$; $h_i, h_{i,j} \in \mathbb{R}$. On one hand, σ_α^i refers to the tensor product of $N-1$ identity matrices of size 2×2 (σ_1) and the Pauli matrix σ_α at position i. On other hand, $\sigma_\alpha^i \alpha_\beta^j$ refers to the tensor product of $N-2$ matrices (σ_1) and σ_α at position i and σ_β at position j. For example, for $N = 3$, $\sigma_z^1 = \sigma_1 \sigma_z \sigma_1$, and $\sigma_z^0 \sigma_x^2 = \sigma_z \sigma_1 \sigma_x$. We can rewrite H as a sum of K 'elementary' Hamiltonians $H_k, k = 1, ..., K$ formed by a matrix of the type σ_α^i or $\sigma_\alpha^i \sigma_\beta^j$ multiplied by a real value.

$$H = \sum_{i,\alpha} h_i \sigma_\alpha^i + \sum_{i,j,\alpha,\beta} h_{i,j} \sigma_\alpha^i \sigma_\beta^j \tag{4}$$

The VQE for this Hamiltonian H works by minimizing (with a classical optimizer) the value of the function (5) where the calculation of $\langle \boldsymbol{\phi(\theta)} | H_k | \boldsymbol{\phi(\theta)} \rangle$ is performed by the quantum computer each time the function (5) is to be evaluated.

$$f(\theta) = \frac{1}{K} \langle \boldsymbol{\phi(\theta)} | H | \boldsymbol{\phi(\theta)} \rangle = \frac{1}{K} \sum_{k=1}^{K} \langle \boldsymbol{\phi(\theta)} | H_k | \boldsymbol{\phi(\theta)} \rangle \tag{5}$$

B. CONDITIONAL VALUE AT RISK - VARIATIONAL QUANTUM EIGENSOLVER (CVARVQE)

The VQE explained above seeks to reduce the mean (5). This would be the original approach of the algorithm, however, it is possible to consider other criteria when forming our asset portfolio.

Barkoutsos et al. [7] argue that an algorithm that seeks a good approximation to the optimum we are looking for is better than an algorithm that tries to reach the optimum and ends up giving a very poor approximation. Since the result that an optimization algorithm gives is usually not the optimal one, but rather the best result found, instead of choosing an objective function that tries to find the optimum, it is proposed to consider an objective function that allows us to find a better result.

With this idea, a better objective function for the optimization problem would be to find the minimum of a set of observations, $\min\{x_1(\theta), ..., x_K(\theta)\}$. However, this objective function is not adequate, since it is very poorly conditioned. It is proposed to minimize Conditional Value at Risk (CVaR) instead.

Formally, the Value at Risk (VaR) of a variable X for a value $\alpha \in \{0, 1\}$ is defined as the smallest value x that verifies $F_X(x) \geq \alpha$ where F_X is the distribution function of the variable. And the CVaR is defined as the expected value, among the observations below the VaR. That is,

$$\text{CVaR}_\alpha(X) = E[X | X \leq F_X^{-1}(\alpha)].$$

When we measure the qubits in a quantum circuit, the result is not a string of bits, but several strings accompanied by a probability. If we assume that $x_1(\theta), ..., x_r(\theta)$ are the observations obtained in a measurement with probabilities $p_0, ..., p_r$, ordered in such a way that $p_i \leq p_{i+1}$, our objective function for a fixed α will be

$$\text{CVaR}_\alpha(\theta) = \frac{1}{\lfloor r\alpha \rfloor} \sum_{i=0}^{\lfloor r\alpha \rfloor} x_i(\theta),$$

where $\lfloor . \rfloor$ is the integer part function.

Using the CVaR as the objective function in our problem becomes a generalization between the objective function $f(\theta)$ of the previous VQE and $\min\{x_1(\theta), ..., x_K(\theta)\}$, since when $\alpha = 1$, $\text{CVaR}_1(\theta) = E[f(\theta)]$ and $\lim_{\alpha \to 0^+} \text{CVaR}_\alpha(\theta) = \min\{x_1(\theta), ..., x_K(\theta)\}$.

VIII. PROPOSED METHODOLOGY

To compare and analyze the algorithms, we have performed several tests. The first test group analyzes the execution times of quantum and classical algorithms. The most important factor to consider is scalability. We examine the possibilities of current large real quantum computers from IBM, identifying the strengths and the limiting factors. In the second set of tests, we check which algorithm provides the best results, based on backtesting. To obtain statistically significant results, we make hundreds of backtesting comparison experiments with different random subsets of stocks from IBEX35 (benchmark stock market index of the Bolsa de Madrid, Spain's principal stock exchange.) [27]

Currently, quantum computers work with a queuing system in which users send their jobs to a quantum computer and wait for the computer to finish executing all the previous circuits, and then receive the results. Unfortunately, this way of working, when coupled with an iterative algorithm such as VQE, which sends a new request to a computer at each iteration, means that the execution of the program is prolonged. Not only because we have to wait in line, but because although the execution of the quantum circuit is fast, the communication times between the quantum computer and our own computer are not. For this reason, it has been decided to execute a VQE on real quantum computers only for validation and illustrative purposes and make the backtesting using simulators.

IX. IMPLEMENTATION OF PORTFOLIO ALGORITHMS

The algorithms mentioned in VI and VII have been implemented in Python using the Qiskit [28] library with different sets of historical adjusted closing price data from the IBEX35 stock market. These data have been taken from Yahoo! Finance [29] using the Python library *bt* [30].

Given a set of assets, we can build a portfolio by associating each asset with a (non-negative) weight that will represent

the proportion of total capital that has been invested in the aforementioned asset. Associated with a portfolio, there are the concepts of profitability and risk (sometimes we will refer to it as volatility) [22]. If we represent by μ the vector of expected returns of the considered assets and by Σ the matrix of variances and covariances between the assets, the return of the portfolio C with weights w will be given by the expression

$$\mu_C = \boldsymbol{\mu}^T w,$$

and its risk for

$$\sigma_C = w^T \Sigma w.$$

In this study, we are going to work exclusively with portfolios whose assets all have the same weight. In other words, the decision that interests us is to choose which assets are interesting to invest in and which are not. We will also impose the restriction that you can only invest in a maximum number of assets (budget). One way of expressing this in a matrix is to replace the vector of weights w by a binary vector x that verifies

$$1^T x = B, \tag{6}$$

where B is the budget. Another approach is to use the vector of weights w and then pick a number B of assets randomly from among those with positive weight. We are indirectly assuming that all assets have the same cost and that with the value B together with (6), we have capital that we want to fully invest.

The classical algorithms used to solve this problem, generally are not comparable. It is not possible to asevere completely that one gives better results than others. However, all of these algorithms become inefficient since they require a large computational cost that grows exponentially with the number of assets considered for the construction of the portfolio.

A. CLASSICAL ALGORITHMS

1) Moving Average Strategy

Following this strategy, the MA of the data is calculated and the assets whose closing price is higher than their moving average are considered. Finally, the desired amount of assets is chosen randomly among them to form the investment portfolio, weighted with equal weights.

The SMAS algorithm has been implemented with a time window of 50 days since most of the tests carried out have been done using similar timeframes.

2) Risk-rentability Optimization

To calculate the efficient frontier, we must solve for each value of R the problem (7).

$$\begin{aligned} &\min_{w \in \mathbb{R}^n} w^T \Sigma w \\ &\text{s.t.} \\ &1^T w = 1, \\ &w_i \geq 0 \; \forall i, \\ &w\boldsymbol{\mu} = R. \end{aligned} \tag{7}$$

The previous problem has a solution for each expected return of the portfolio, R, fixed. We obtain for each valid R a point of the efficient frontier. Any efficient portfolio is a good solution for this type of optimization algorithm. There are multiple criteria for choosing the most convenient efficient portfolio for the investor, and there are lines of study focused exclusively on this aspect.

Following the MVO algorithm, some efficient frontier portfolios are calculated and one of them is chosen using the cvxopt solver [31].

3) Sharpe Ratio Optimization

The problem of maximizing the Sharpe Ratio is given by

$$\begin{aligned} &\max_{w \in \mathbb{R}^n} \frac{\boldsymbol{\mu}^T w - R}{\sqrt{w^T \Sigma w}} \\ &\text{s.a} \\ &1^T w = 1, \\ &w_i \geq 0 \; \forall i. \end{aligned} \tag{8}$$

To solve the Sharpe Ratio maximization problem, the SRO has been implemented using function *calc_mean_var_weights* from package *ffn* (complement of the *bt* Python library).

B. QUANTUM ALGORITHMS

The formulation of the portfolio problem that we solve with the VQE algorithm is (9), where x will represent the binary vector of assets chosen to build our portfolio, $\boldsymbol{\mu}$ and Σ the return vector and the matrix of variances and covariances of the n assets considered, B the number of assets in which one wants to invest (budget) and q a positive term that measures the risk aversion of the investor.

$$\begin{aligned} &\min_{x \in \{0,1\}^n} q x^T \Sigma x - \boldsymbol{\mu}^T x \\ &\text{s.a} \\ &1^T x = B. \end{aligned} \tag{9}$$

The expression $1^T x = B$ is transformed into $(1^T x - B)^2$ and subtracted from the objective function of (9) multiplied by a penalty term. It must be taken into account that by performing this transformation we accept that the solution found chooses a number of assets different from the budget. The penalty term is the one that will measure how much it is acceptable to relax this restriction. For example, if it is very detrimental to our interests when solving the problem, the penalty term will have to be greatly increased. By operating, we transform the expression in (9) into a quadratic form (10), for certain A and b.

$$x^T A x + b^T x, \; x \in \{0,1\}^n \tag{10}$$

Performing the change of variable $x_i = (1 - z_i)/2$, for $z_i \in \{-1, 1\}$ in (10), we transform our problem into a new problem, which will consist of minimizing the expression (11), which coincides with the expression of the energy of the Hamiltonian of the Ising model. That is, our problem is

FIGURE 2. Example of full entangled ansatz VQE Quantum Circuit for three assets.

equivalent to finding the lowest energy level of a Hamiltonian, which is possible using the VQE, where C and d are the coefficients in the new domain.

$$z^T C z + d^T z, \; z \in \{1, -1\}^n \tag{11}$$

The Hamiltonian of the Ising model is constructed by substituting in (11) the terms z_i by σ_z^i and the terms $z_i z_j$ by $\sigma_z^i \sigma_z^j$, obtaining a decomposition of the Hamiltonian as shown (4). Using Qiskit functions, we transform our problem into this new formulation in terms of Pauli Z-matrices. This formulation is common to both VQE and CVaRVQE.

The VQE algorithm has a quantum part and a classical part. The classical optimizer performs several iterations to calculate the parameters that optimize the problem and executes a circuit on a quantum computer in each iteration. Furthermore, for our type of problem, the use of the COBYLA optimizer [6] is recommended. An example of a quantum circuit that VQE uses for three assets can be seen in Fig. 2.

The execution time of a job in a quantum computer can be divided into communication time (between the classical computer and the quantum computer), validation time (verification of the sent job), queue time, and execution time. This last time comprises the time it takes to execute the circuit in the quantum chip and the measurements and post-processing that are made to transform the values of the qubits into bits understandable by a classical computer. It must be taken into account that this execution time includes several executions of the quantum circuit and the corresponding measurements of the qubits after each execution of the circuit, in order to obtain the probabilities of the possible solutions.

In general, the validation time is negligible, being only a few milliseconds. In addition, the running time of a quantum circuit is also extremely short, so the running time is mainly the measurement time of the qubits. Finally, we comment that after carrying out various tests with real backends with different job numbers in the queue, it has been observed that communication times do not significantly affect the total execution time of the job. As such, queue time is by far the most time-consuming, but the analysis we are really interested in this section is execution time.

The stability of the current qubits does not reach the second [32], so the execution time of a quantum circuit (each shot) must not exceed that time in any case. With this in mind, we run the following tests to assess the possibilities of using this quantum portfolio optimization strategy on the largest available IBM real quantum computers.

During the preparation of this work (December 2022), we utilized the most stable and proven 27 qubit machines from the Falcon family and the bigger experimental 127 qubit machines from the Eagle family.

C. CONSIDERATIONS FOR 27 QUBIT MACHINES

Falcon quantum computers have a heavy hexagonal layout. This has to be taken into account to define a hardware viable ansatz. For this experiment, we selected *ibm_cairo* as a 27 qubit machine (December 2022). The processor type is Falcon r5.11 (Multi-chip stack: The qubit chip is bump-bonded to an "interposer" chip that provides readout and signal delivery wiring). It has a Quantum Volume (QV) of 64 and can perform 2.4K Circuit Layer Operations Per Second (CLOPS).

The system has the following native basis gates: CX, ID, IF_ELSE, RZ, SX, X.

Its median errors are:

- Median CNOT Error: $1.049\text{e}{-2}$
- Median Readout Error: $1.180\text{e}{-2}$
- Median T1: 106.81 μs
- Median T2: 102.88 μs

The chip has the individual error distribution shown in Fig. 3. Observe that not all the qubits and connections are equally reliable, with bigger error rates on lighter-colored ones.

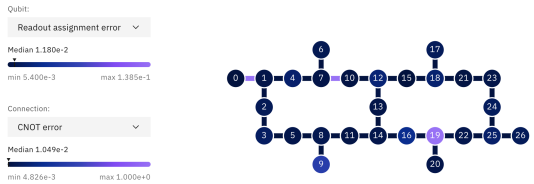

FIGURE 3. Layout and individual error distribution on *ibm_cairo"* real quantum computer.

Before executing the quantum circuits we have to map the circuit qubits to the real hardware one, and also translate the original gates to the equivalent native ones.

To get the best results in the hardware we need to consider two important aspects: first, we need to use the qubits and links with better quality, and second, we need to select the mapping of the qubits to maximize the CNOT gates between real qubits with actual connection. If we need to perform CNOT gates between non-conected qubits, we will need to add swap gates to move the needed qubits to a position in the chip with actual connection, generating a larger circuit.

We need to maintain the depth of the circuit to the minimum to avoid errors and to maintain our execution within the chip coherence time. If the errors grow up, the expectation values calculated became nearly random, and the classical optimizer would be unable to converge.

Avoiding qubits and CNOT links with bigger errors, we can find a linear path with a good performance of 19 qubits, as shown in Fig. 4.

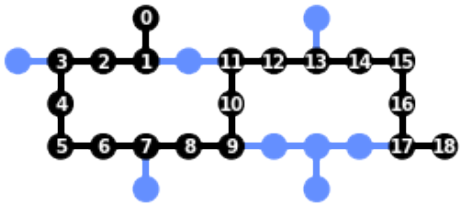

FIGURE 4. Optimum linear path with 19 qubits on "ibm_cairo" real quantum computer.

We can create a VQE algorithm to select the best 5 out of 19 assets using this quantum computer. Randomly selecting 19 stocks from IBEX35 we get:

ELE.MC, ITX.MC, FER.MC, CLNX.MC,
ANA.MC, ENG.MC, IBE.MC, BBVA.MC,
ACS.MC, VIS.MC, MTS.MC, GRF.MC,
COL.MC, MEL.MC, ACX.MC, NTGY.MC,
TEF.MC, SGRE.MC, SAN.MC.

From Yahoo! Finance, we obtain the closing stock price for two months, from 2022-10-1 to 2022-12-1. The evolution of the stock prices can be seen in Fig. 5.

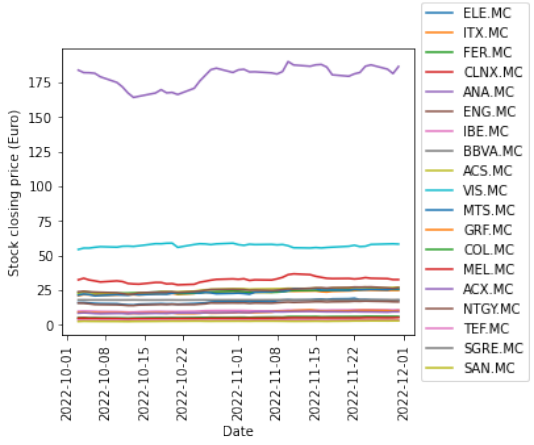

FIGURE 5. Evolution of closing price for the selected 19 stocks from IBEX35.

To make the optimization we create an ansatz with linear entanglement that fits on the selected linear path and uses one qubit for each of the 19 stocks (Fig. 6).

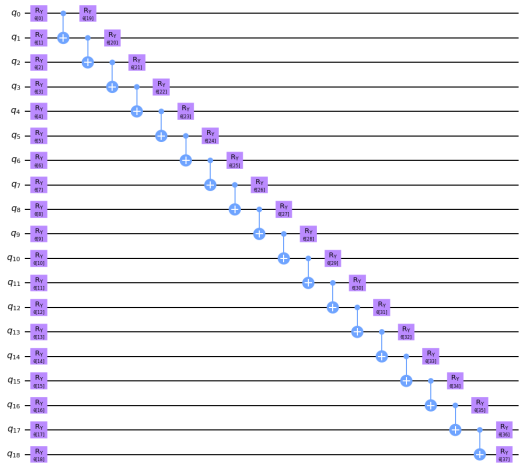

FIGURE 6. Linear entangled ansatz VQE Quantum Circuit for 19 assets.

D. CONSIDERATIONS FOR 127 QUBIT MACHINES

Eagle quantum computers have also a heavy hexagonal layout. This has to be taken into account to define a hardware viable ansatz. For this experiment, we selected *ibm_washington"* as a 127 qubit machine (December 2022). The processor type is Eagle r1 (increased scalability through novel packaging techniques). It has a QV of 64 and can perform 850 CLOPS.

The system has the following Basis gates: CX, ID, IF_ELSE, RZ, SX, X.

Its median errors are:

- Median CNOT Error: $1.378\text{e}{-2}$
- Median Readout Error: $1.080\text{e}{-2}$
- Median T1: 99.55 μs
- Median T2: 93.66 μs

The chip has the individual error distribution shown in Fig. 7.

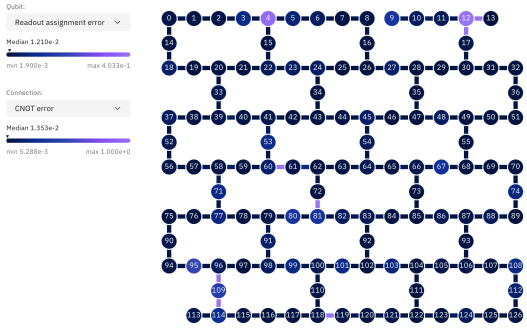

FIGURE 7. Layout and individual error distribution on *ibm_washington"* real quantum computer.

Avoiding qubits and CNOT links with bigger errors, we can find a circular path with an optimum performance of 28 qubits, as shown in Fig. 8. It is possible to find longer paths, but currently, this one will avoid an ansatz with depth near the coherence time of the chip.

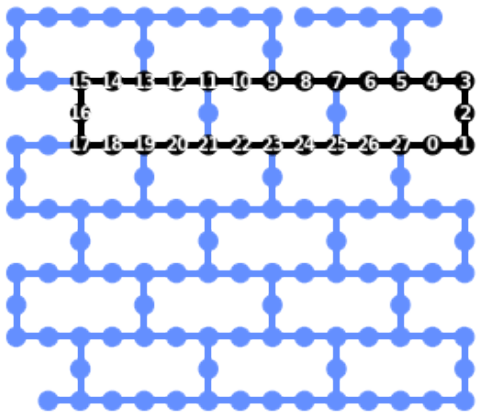

FIGURE 8. Optimum circular path with 28 qubits on "$ibm\_washington$" real quantum computer.

We can create a VQE algorithm to select the best 5 out of 19 assets using this quantum computer. In order to do this, we create an ansatz with circular entanglement (Fig. 9).

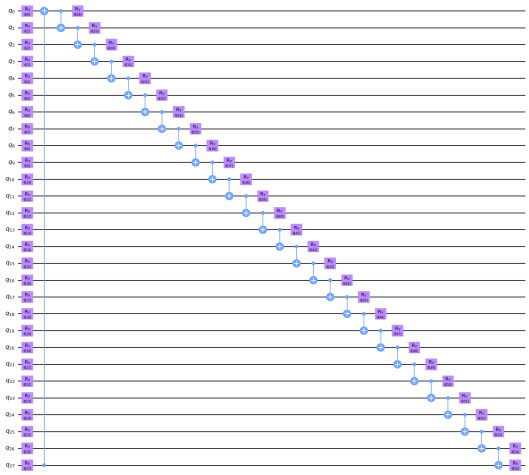

FIGURE 9. Linear entangled ansatz VQE Quantum Circuit for 19 assets.

X. RESULTS

A. BASIC EXECUTION 3 ASSETS ON REAL QUANTUM COMPUTER VS CLASICAL

Figure 10 represents the obtained probabilities after executing a VQE with the data of three random assets from October 2020. For the execution of this instance, the 5-qubit real backend *ibmq_athens* has been used, simply because it was the quantum computer with fewer tasks in the queue among those that meet the minimum requirements for the problem at that moment. In addition, the number of iterations of the VQE has been limited to 20.

Table 1 shows the exact values of the objective function for each possible choice of assets chosen. As can be seen, the result obtained with the algorithm is different from the optimal one, this is because 20 iterations are too few for the optimizer to find the desired minimum in this case and configuration

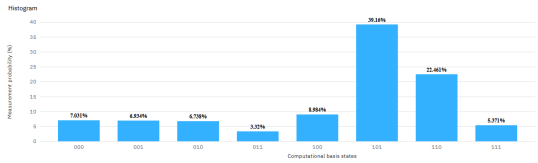

FIGURE 10. Probabilities obtained with VQE after 20 iterations on a real quantum computer (*ibmq_athens*).

TABLE 1. Results obtained with the three asset example using NumPyMinimumEigensolver.

Choice	[111]	[011]	[101]	[001]
Values	4.0006	16.0000	16.0005	36.0000
Choice	[110]	[010]	[100]	[000]
Values	16.0001	35.9996	36.0004	64.0000

B. EXECUTION TIME ON REAL QUANTUM COMPUTER VS CLASICAL

In this example, the approximate average time that each iteration has taken is 2 hours (including communication and queue times), however, these data is not really relevant, as it depends on many factors unrelated to the algorithm. As a comment and by way of illustration, it must be said that on other occasions, circuits with twice as many qubits have taken around 44 seconds for each iteration. The big difference is due to the waiting time in the queue. In this case, the execution time in each iteration of the algorithm has been around 9 seconds.

Although this time may seem quite long, the advantages of quantum computers over classical ones are that, presumably, this execution time does not increase exponentially with the number of assets (qubits), which is the case with classical computers. To verify experimentally that this statement is correct, a VQE has been executed with data sets with between 2 and 15 random assets and the average execution time has been calculated in each case. These executions were carried out on the backend *ibmq_16_melbourne* (real quantum computer with only 15 qubits. Despite its name, it had only 15 usable qubits due to error rate of the qubit 16). The results are shown in Fig. 11.

To compare with the behavior of classical algorithms, the same experiment could be carried out as with the quantum, however, working with 35 assets (Fig. 12). In the case of the SMAS, its computational cost is effectively linear, since the calculation of the MA is of np sums and p products, where p is the number of periods considered and n the number of assets with those who work. As for SRO and MVO, both algorithms involve solving a quadratic programming problem, which is difficult to solve when real data is used together with extra constraints [33]. Both algoritms show an exponential trend. As the numbers grow larger, the exponential nature

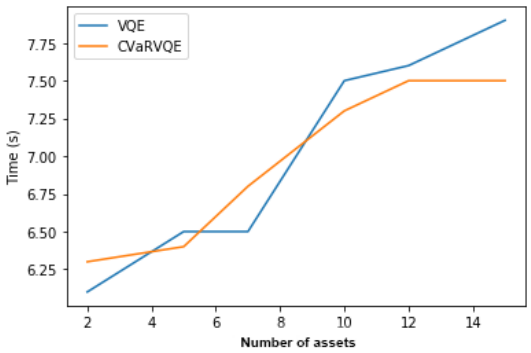

FIGURE 11. Execution time for Quantum Algorithms depending on the number of assets on *ibmq_16_melbourne* (December 2022).

of the algorithms becomes more apparent (not shown in the figure to maintain the scale of quantum and classical experiment results).

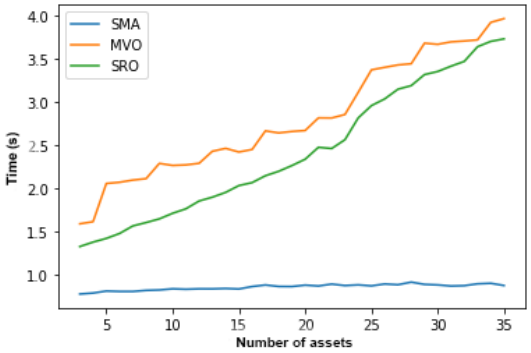

FIGURE 12. Execution time of classical algorithms depending on the number of assets.

An alternative to using a real quantum computer is the use of a simulator. These simulators mimic the behavior of a quantum computer, making them very inefficient, but with the advantage of not having to wait in line or have the problem of communication times. In the previous example, we saw that the time for each iteration was huge, so the same problem has been solved with the simulator *ibmq_qasm_simulator* by allowing the optimizer to perform 100 iterations. The results are seen in Fig. 13. In this case, the desired optimum has been reached.

C. EXECUTING ON 27 QUBIT MACHINES

Fig. 14 shows the previously defined ansatz circuit transpiled to the system's available gates and mapped to the selected 19 qubits, having a depth of 27 Circuit Layer Operations on the real hardware. This takes a time of approximately 1125 μs on the quantum chip for each shot. The actual execution for 100000 shots takes an average of 35.3 s in the quantum system (including quantum compute and near-time classical pre- and post-processing. The queue time is not included.)

The CVaR VQE algorithm converges very quickly and

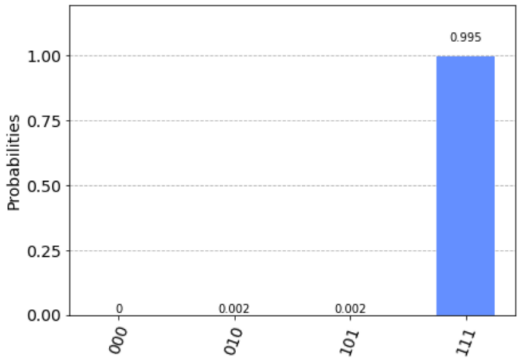

FIGURE 13. Probabilities obtained with VQE after 100 iterations on a simulator.

FIGURE 14. Linear entangled ansatz VQE Quantum Circuit for 19 assets transpiled for "ibm_cairo" real quantum computer.

after 10 iterations (Fig. 16) we obtain the following proposed optimal portfolio:

ITX.MC, BBVA.MC, ACS.MC, ACX.MC, SAN.MC

As an example, Fig. 15 shows the probabilities measured in the real quantum computer for one VQE iteration executed on *ibm_cairo* with 100000 shots.

We can now use the operator we built above regardless the specifics of how it was created. We set the algorithm for the NumPyMinimumEigensolver so we can have a classical reference. A backend is not required since this is computed classically not using quantum computation. It took 126.57s to obtain the following results:

ITX.MC, BBVA.MC, ACS.MC, MTS.MC, SAN.MC

As can be observed, it differs from the quantum result only in one of the selected stocks.

D. EXECUTING ON 127 QUBIT MACHINES

As shown in Fig. 17, this circuit transpiled to the system available gates and mapped to the selected 28 qubits has a

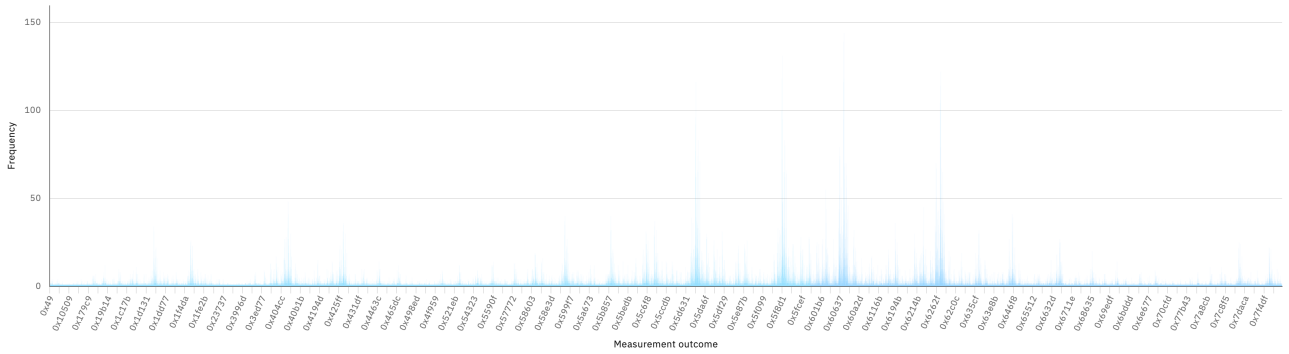

FIGURE 15. Example of probabilities measured on *ibm_cairo"*, 27 qubit real quantum computer.

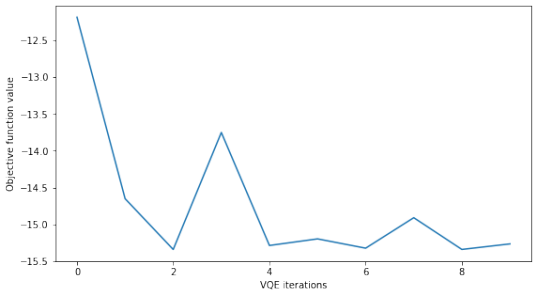

FIGURE 16. Convergence of CVaR VQE on *ibm_cairo"*, 27 qubit real quantum computer.

depth of 37 Circuit Layer Operations on the real hardware. This will take an estimated time of 4352 μs on the quantum chip. The actual execution for 100000 shots takes 48.6 s in the quantum system. The result is too big to be displayed as a histogram.

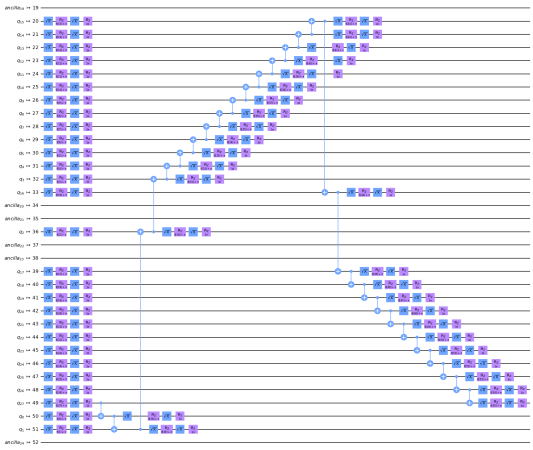

FIGURE 17. Linear entangled ansatz VQE Quantum Circuit for 19 assets transpiled for "$ibm\_washington$" real quantum computer.

E. BACKTESTING

To analyze the performance of the algorithms, we prepare an experimental set with IBEX35 historical data from 2016 to 2020. As many algorithms rely on historical data, we left 2016 data for historical calculations, and start investing in 2017. Each strategy is permitted to calculate and rebalance the portfolio once a month. To be able to compare the results, the five tested algorithms (SMA, MVO, SRO, VQE, and VaR-VQE) were configured under the same conditions of risk aversion factor and avoided selecting 2 out of 5 available stocks. Then the portfolio is rebalanced with the selected stocks equally weighted.

A suite of 210 backtesting experiments has been carried out with 5 different random assets from IBEX35. Each experiment provides a backtesting comparison of the performance of the different algorithms with the same data. An example of the results obtained on each experiment can be shown in Fig. 18.

The statistics of the results have been collected in tables 2 and 3 where we can observe the mean and variance of the results of each algorithm among the 210 simulations, as well as the number of times that each algorithm has given a better solution than the others. The high variance reflects the unpredictability of the stock market, where past events do not guarantee future outcomes.

TABLE 2. Main statistics obtains after the execution of 210 simulations with six random assets.

	SMAS	SRO	MVO	VQE	CVaRVQE
Mean	0.263693	0.336095	0.388857	0.320283	0.311016
Var.	0.343655	0.320120	0.312191	0.299983	0.337807

TABLE 3. Number of times that the algorithm gets the best strategy (210 simulations with six random assets).

	SMAS	SRO	MVO	VQE	CVaRVQE	Total
Times Best	29	46	50	40	45	210

SMAS performed poorly as expected, since it is a naive approach. We included SMAS as a baseline to compare the

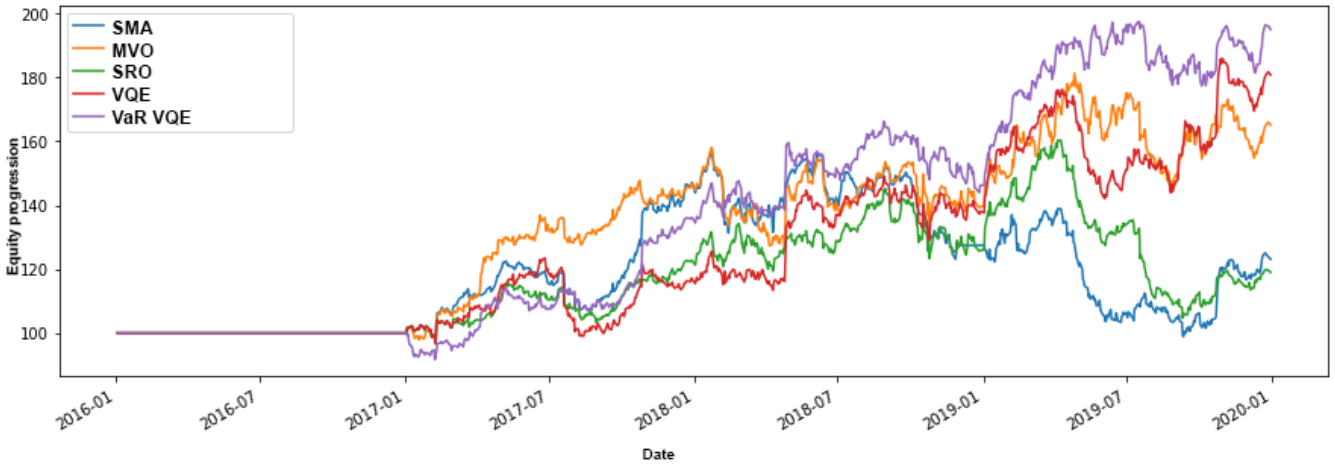

FIGURE 18. Example of one iteration of backtesting comparison of portfolio optimization with 5 random assets from IBEX35.

other algorithms, both quantum and classic. Given that the rest of the algorithms have a very similar mean and a very large variance, it has been decided to filter the 210 executions based on the mean and variance of each execution.

We want to find out which algorithm performs better for a given set of assets. To do this, we need to compare the results of each algorithm for each record. A record is a set of stocks that we run the algorithms on. We use the intra-record variance to quantify how much the results differ from each other for each record. A higher variance means that the results are more spread out and less similar. We are interested in the records that have a high variance, because they show us which algorithm stands out from the others. To filter out the low-variance records, we calculate the average of all the variances for all the records. Then we only keep the records that have a variance higher than this average. These are the records that we use to compare the algorithms and find out which one is better. We check again how many times each algorithm has provided the best of the strategies. The results are collected in table 4.

TABLE 4. Number of times that the algorithm gets the best strategy (filtered simulations).

	SMAS	SRO	MVO	VQE	CVaRVQE	Total
Times Best	11	12	14	14	21	72

As can be seen, although initially, the MVO seemed to give better results on more occasions than the other algorithms, in particular the quantum ones, after filtering we verified that the results of the quantum algorithms are much better than the others on more occasions, highlighting especially the VQE-CVaR. In any case, all the algorithms (both before and after filtering) always result in a better strategy than the others between 20% and 30% of the time, it is challenging to determine with complete certainty which algorithm outperforms the others.

The results shown in tables 3 and 4, suggest that both the VQE and the CVaRVQE are useful algorithms and to some extent competitive with the classical algorithms.

But the real advantage of using quantum computers for the portfolio optimization problem is not that the algorithm provides better or worse results, but that for a tremendously large number of assets, the classical algorithms become unfeasible. However, quantum algorithms would not have that problem. As long as there is a quantum computer with sufficient specifications, the VQE, and VQE-CVaR algorithms will allow us to calculate a good solution to the portfolio optimization problem in a reasonable time.

XI. CONCLUSION

This work addresses a backtesting methodology of classical and quantum computing algorithms for portfolio optimization. Results from backtesting the VQE and CVaR-VQE algorithms suggest that they are useful and competitive with classical algorithms to some extent.

The scalability of current quantum computers from IBM is analyzed by testing their execution times. A second set of tests checks which algorithm provides the best results based on backtesting using historical data from the IBEX35 stock market in Spain. The algorithms were implemented using Python and the Qiskit library.

For running on real quantum computers, we take into account the hardware design and qubit error distribution to find the best qubit mappings and gates to use. We also aim to keep the depth of the quantum circuits to a minimum to minimize errors and maintain execution within the coherence time of the chip. It has been used IBM backends of 27 and 127 qubits.

The backtesting results showed that, after filtering the experiments, both VQE and CVaR-VQE algorithms were competitive with the classical algorithms, with the advantage of being able to handle a large number of assets in a reasonable time on a future quantum computer.

Our models are still limited by the assumptions and simplifications that we have made to represent the stock market as a quantum system. Therefore, as future work, we plan to develop more realistic models that can capture the complexity and uncertainty of the real stock market conditions. The main problem to solve is how to select a portfolio of securities that maximizes the expected return and minimizes the risk, while taking into account the transaction costs and the integer constraints on the quantities of securities. Transaction costs are the fees that investors have to pay when buying or selling securities, such as commissions, taxes, or bid-ask spreads. Integer constraints are the restrictions that some securities can only be bought or sold in fixed units, such as shares or bonds. These factors make the portfolio selection problem more realistic but also more challenging to solve. We also plan to implement error mitigation techniques to reduce the noise and errors that affect the quantum algorithms. Moreover, we plan to test our models on larger and more diverse data sets to validate their scalability and robustness.

ACKNOWLEDGMENT

This work was supported by grant PID2021-123041OB-I00 funded by MCIN/AEI/ 10.13039/501100011033 and by "ERDF A way of making Europe" and by the CM under grant S2018/TCS-4423.

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Quantum team.

REFERENCES

[1] M. M. Waldrop, "The chips are down for Moore's law," Nature News, vol. 530, p. 144, Feb. 2016. Cg_type: Nature News Section: News Feature.
[2] R. P. Feynman, "Simulating Physics with Computers," International Journal of Theoretical Physics, vol. 21, no. 6, pp. 467–488, 1982.
[3] E. Yndurain, S. Woerner, and D. J. Egger, "Exploring quantum computing use cases for financial services," IBM Institute for business value. URL: https://www. ibm. com/downloads/cas/2YPRZPB3.[dl: 29.07. 2020], 2019.
[4] J. Gacon, C. Zoufal, and S. Woerner, "Quantum-Enhanced Simulation-Based Optimization," in 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), pp. 47–55, 2020.
[5] R. O. Michaud and R. O. Michaud, Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation. Oxford University Press, Mar. 2008. Google-Books-ID: CzhnDAAAQBAJ.
[6] Qiskit, "Simulating Molecules using VQE." https://community.qiskit.org/textbook/ch-applications/vqe-molecules.html.
[7] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, "Improving Variational Quantum Optimization using CVaR," Quantum, vol. 4, p. 256, Apr. 2020. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.
[8] M. J. Best, Portfolio optimization. CRC Press, 2010.
[9] M. E. Stucke, "Behavioral economists at the gate: Antitrust in the twenty-first century," Loy. U. Chi. LJ, vol. 38, p. 513, 2006. Publisher: HeinOnline.
[10] S. K. Mehra, "Competition Law for a Post-Scarcity World," Tex. A&M L. Rev., vol. 4, p. 1, 2016. Publisher: HeinOnline.
[11] S. A. Vavasis, "Quadratic programming is in NP," Information Processing Letters, vol. 36, no. 2, pp. 73–77, 1990. Publisher: Elsevier.
[12] A. J. King and D. L. Jensen, "Linear-quadratic efficient frontiers for portfolio optimization," Applied Stochastic Models and Data Analysis, vol. 8, no. 3, pp. 195–207, 1992. Publisher: Wiley Online Library.
[13] M. J. Best and J. K. Kale, "Quadratic programming for large-scale portfolio optimization," in Financial Services Information Systems, pp. 531–548, Auerbach Publications, 2000.
[14] D. A. Fedorov, B. Peng, N. Govind, and Y. Alexeev, "VQE method: A short survey and recent developments," Materials Theory, vol. 6, no. 1, pp. 1–21, 2022. Publisher: SpringerOpen.
[15] J. Sen and A. Dutta, "A comparative study of hierarchical risk parity portfolio and eigen portfolio on the NIFTY 50 stocks," in Computational Intelligence and Data Analytics, pp. 443–460, Springer, 2023.
[16] A. Dezhkam and M. T. Manzuri, "Forecasting stock market for an efficient portfolio by combining XGBoost and Hilbert-Huang," Engineering Applications of Artificial Intelligence, vol. 118, p. 105626, 2023. Publisher: Elsevier.
[17] C. Garcia and F. Gould, "Survivorship bias," Journal of Portfolio Management, vol. 19, no. 3, p. 52, 1993. Publisher: Pageant Media.
[18] G. Daniel, D. Sornette, and P. Woehrmann, "Look-ahead benchmark bias in portfolio performance evaluation," The Journal of Portfolio Management, vol. 36, no. 1, pp. 121–130, 2009. Publisher: Institutional Investor Journals Umbrella.
[19] M. Owhadi-Kareshk and P. Boulanger, "Portfolio Optimization on Classical and Quantum Computers Using PortFawn," arXiv preprint arXiv:2112.08998, 2021.
[20] M. T. Faber, "A Quantitative Approach to Tactical Asset Allocation," The Journal of Wealth Management, vol. 9, p. 69, Jan. 2007.
[21] M. I. Ahmed, W. Ghohui, M. Hasan, A. Sattar, M. Ahmed, and R. U. Rehman, "Performance of Moving Average Investment Timing Strategy in UK Stock Market: Individual Stocks versus Portfolios," Journal of Economic and Social Studies, vol. 7, no. 2, 2018.
[22] H. Markowitz, "Portfolio Selection*," The Journal of Finance, vol. 7, no. 1, pp. 77–91, 1952. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1952.tb01525.x.
[23] H. Markowitz, Portfolio Selection: Efficient Diversification of Investments. Yale University Press, 1959.
[24] W. F. Sharpe, "Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk*," The Journal of Finance, vol. 19, no. 3, pp. 425–442, 1964. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1964.tb02865.x.
[25] D. H. Bailey and M. Lopez de Prado, "The Sharpe ratio efficient frontier," Journal of Risk, vol. 15, no. 2, p. 13, 2012.
[26] S. Farinelli, M. Ferreira, D. Rossello, M. Thoeny, and L. Tibiletti, "Beyond Sharpe ratio: Optimal asset allocation using different performance ratios," Journal of Banking & Finance, vol. 32, no. 10, pp. 2057–2063, 2008. Publisher: Elsevier.
[27] "BME Bolsas y Mercados Españoles." https://www.bolsasymercados.es/esp/Home.
[28] "Qiskit: An Open-source Framework for Quantum Computing," 2017. https://qiskit.org/.
[29] Yahoo Finanzas, "Componentes de IBEX 35... (^IBEX)." https://es.finance.yahoo.com/quote//%5EIBEX/components?p=\%5EIBEX.
[30] P. Morissette, "bt - Flexible Backtesting for Python," 2015. https://pmorissette.github.io/bt/.
[31] J. Dahl, L. Vandenberghe, and M. Andersen, "CVXOPT - Python Software for Convex Optimization," 2004. https://cvxopt.org/.
[32] J. Gambetta and S. Sheldon, "Cramming More Power Into a Quantum Device | IBM Research blog," 2019.
[33] R. Mansini and M. G. Speranza, "An exact approach for portfolio selection with transaction costs and rounds," IIE transactions, vol. 37, no. 10, pp. 919–929, 2005. Publisher: Taylor & Francis.

GINÉS CARRASCAL was born in Salamanca, Spain in 1975. He received an M.Sc. degree in physics from the University of Salamanca (Spain) in 1999.

Since 2000 he has been working as Architect at IBM Consulting Spain, getting involved with quantum computing in 2017, and now acting as Quantum Technical Ambassador and Qiskit Advocate. He has become IBM Certified Associate Developer - Quantum Computation using Qiskit v0.2X in 2021. Since 2014 he has been an Adjunct Professor of computer science at Universidad Carlos III de Madrid and since 2018, an Adjunct Professor at Universidad Complutense de Madrid at the computer science departmental section of the Mathematics Faculty. His research interest includes Artificial Intelligence and the application of Quantum Computing to optimization problems, especially but not only in the field of Banking and Financial Services.

Prof. Ginés Carrascal has received two Outstanding Technical Achievement Awards (OTAA), the highest IBM technical award in 2012 and 2016.

PAULA HERNAMPEREZ was born in Valladolid, Spain in 1996. She received her M.Sc degree in Mathematical Engineering from the Complutense University of Madrid (Spain) in 2021.

Since 2019, she works as Researcher at the CARTIF Technology Centre's Energy Division, and as Team Leader since 2022, participating in national and international projects related to climatic change prevention, new battery technologies development, energy efficiency, optimal control of smart grids and renewable energy sources penetration. Her current research interests include Machine learning, Optimization problems resolution new methodologies and Data Sciences, and the application of new technologies to the field of energy and energy markets.

GUILLERMO BOTELLA (SM'19) received the M.Sc. degree in Physics (Fundamental) in 1998, the M.Sc. degree in Electronic Engineering in 2001 and the Ph.D. degree (Computer Engineering) in 2007, all from the University of Granada, Spain.

He was a research fellow funded by the EU working at the University of Granada, Spain, and the Vision Research Laboratory at University College London, UK. After that, he joined as an Assistant Professor at the Department of Computer Architecture and Automation of the Complutense University of Madrid, Spain where he is currently an Associate Professor. He has performed research stays acting also as visiting professor from 2008 to 2012 at the Department of Electrical and Computer Engineering, Florida State University, Tallahassee, USA. His current research interests include Image and Video Processing for VLSI, FPGAs, GPGPUs, Embedded Systems, and novel computing paradigms such as analog and quantum computing.

Dr. Botella, since 2019 has become an IEEE Senior Member.

ALBERTO A. DEL BARRIO (SM'19) received the Ph.D. degree in Computer Science from the Complutense University of Madrid (UCM), Madrid, Spain, in 2011.

He has performed stays at Northwestern University, the University of California at Irvine, and the University of California at Los Angeles. Since 2021, he is an Associate Professor (tenure-track, civil-servant) of Computer Science with the Department of Computer Architecture and System Engineering, UCM. His main research interests include Design Automation, Arithmetic, Analog/Quantum Computing, and their application to the field of Artificial Intelligence.

Dr. del Barrio has been the PI of the PARNASO project, funded by the Leonardo Grants program by Fundación BBVA, and currently, he is the PI of the ASIMOV project, funded by the Spanish MICINN, which includes a work package to research on the applicability of Quantum Computing. Since 2019 he is an IEEE Senior Member and since December 2020, he is an ACM Senior Member, too.