Quantum Portfolio Optimization: An Extensive Benchmark

Eric Stopfer[1,*] and Friedrich Wagner[1]

[1]Fraunhofer Institute for Integrated Circuits, Nürnberg

[*]eric.stopfer@iis.fraunhofer.de

September 2025

Abstract

Recently, several researchers proposed portfolio optimization as a potential use case for quantum optimization. However, the literature is lacking an extensive benchmark quantifying the potential of quantum computers for portfolio optimization. In this work, we fill this gap. We provide a computational study, comparing quantum approaches against state-of-the-art classical methods on a meaningful, real-world instance set. In particular, we compare quantum annealing and the quantum approximate optimization algorithm against classical mixed-integer programming, simulated annealing, steepest descent local search, tabu search and a problem-tailored heuristics. We consider a variant of portfolio optimization which we show to be particular difficult for classical solvers in practice. Our benchmark comprises 250 instances with up to 1,000 assets from actual stock data. The results show that all instances can be solved to proven optimality by mixed-integer programming in the order of seconds. Moreover, the problem-tailored heuristic consistently outperforms quantum approaches in terms of solution quality for fixed runtime. Thus, we conclude that there is only very limited room for a potential quantum advantage in portfolio optimization.

1 Introduction

Combinatorial optimization problems play a central role in various fields such as logistics and finance [1]. In practice, these problems are often tackled by mixed-integer programming (MIP) solvers, which can solve even large instances of NP-hard problems to proven optimality [2, 3]. Nevertheless, some practically relevant problems remain intractable for state-of-the-art classical methods [4]. This fact has motivated researchers to develop quantum algorithms for such problems [5]. However, the limitations of existing quantum computers prohibit the implementation of exact quantum algorithms for optimization problems [6]. As a result, researchers focus on heuristic quantum algorithms, which require significantly less resources than exact approaches [7]. Well-known examples of such quantum heuristics are quantum annealing [8, 9] and the quantum approximate optimization algorithm (QAOA) [10, 11]. Recently, several works proposed portfolio optimization as a suitable problem class for achieving quantum advantage [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. In this work, we provide experimental evidence suggesting that even large instances of portfolio optimization can be solved to proven optimality by modern MIP solvers in the order of seconds. Moreover, our problem-tailored classical heuristic outperforms quantum approaches in terms of solution quality for fixed runtime. Our results thus set the bar high for a practical quantum advantage in portfolio optimization.

Related work. Various variants of the portfolio optimization problem exist in literature. The authors of [19, 15, 18, 16] consider a variant that is known as *choose-asset-or-not*. Therein, as the name suggests, the task is to decide for each asset if we add it to the portfolio or not. In this work, we consider an extended variant where the goal is to find optimal *weights* of assets in the portfolio, that is, we assign a fraction of the invested money to each asset. This variant can be further subdivided with respect to the considered objective. Some researchers focus on maximizing return with limited volatility [13, 17]. Others aim to minimize the volatility with a fixed return [18, 23]. Several authors combine both objectives by maximizing a weighted sum of return and volatility [19, 15, 24]. There also exist approaches in literature which include further practical aspects of stock trading into the model, for example, short-selling-options, risk-free investments and costs that are associated with asset acquisitions [12, 24, 20].

Classical solution approaches to the portfolio optimization problem include both heuristics [25, 26, 24, 27] and exact methods [28, 29, 30]. Notably, the heuristic developed in [30] outperforms the commercial MIP solver CPLEX [31] for variance-minimizing choose-asset-or-not instances in terms of solution quality for predefined runtimes.

Recently, researchers proposed quantum approaches to the portfolio problem, including QAOA [19, 20], Variational Quantum Eigensolver (VQE) [22] and quantum annealing [13, 17, 18, 16, 21]. The authors of [13], [17] and [16] conclude that quantum annealing may outperform classical methods for portfolio optimization. However, their studies employ a closed-source hybrid quantum-classical solver, which makes it impossible to divide classical from quantum contributions to the solution. On the contrary, in this work, we stick to methods with known classical and quantum part. Regarding QAOA, the authors of [20] report near-optimal results for small problem instances of 8 assets with an idealized simulator of a gate-based quantum computer. Ref. [19] provides guidelines on the selection of penalty and circuit parameters as well as on the classical optimizer. In [18, 21], the authors use reverse quantum annealing to improve upon conventional quantum annealing. Ref. [22] provides studies the influence of penalty coefficients in VQE as well as the effects of different quantum devices. Other approaches cover problem sparsification [26], problem decomposition [15] and quantum circuit cutting techniques [32]. For a comprehensive review on solution approaches to the portfolio optimization, we refer the interested reader to [33].

Our contribution. In this work, we provide a computational study, including 250 real-world instances with up to $1,000$ assets. We compare QAOA, quantum annealing, two exact MIP solvers, three meta-heuristics and one problem-specific heuristics. Our contribution is threefold. First, the large instance set allows us to draw statistically significant conclusions. Second, applying both state-of-the-art quantum and classical solution methods, we provide a comprehensive and fair benchmark. Third, our methods have well-defined quantum and classical parts such that we can precisely divide quantum from classical contributions.

The remainder of this paper is organized as follows. In section 2, we define three variants of portfolio optimization which we consider in this work. Section 3 studies the classical hardness of these variants to identify the one with the largest potential for quantum advantage. In section 4 we develop a quadratic unconstrained binary optimization (QUBO) model for the most difficult portfolio optimization variatn. Section 5 introduces the solution methods we compare in our benchmark. In Section 6, we report on the results of our benchmark study. Finally, in section 7, we summarize our findings, draw a conclusion and an state open questions.

2   The Portfolio Optimization Problem

In this section, we introduce three different variants of the portfolio optimization problem, which we consider in our computational study. As originally formulated by Markowitz in [34], the portfolio optimization problem asks for a selection of an asset portfolio which maximizes the expected portfolio return while minimizing the return variance, also called *volatility*.

More formally, given a set of n assets, the task is to find optimal *asset weights* $\omega_i \in [0,1]$, $i \in \{1, \ldots, n\}$. The asset weight ω_i defines the share of the invested money allocated to the i-th asset. Accordingly, it holds

$$\sum_{i=1}^{n} \omega_i = 1 \ .$$

We impose upper bounds $u_i \in (0,1]$ on the asset weights,

$$\omega_i \leq u_i \quad \forall i \in 1, \ldots, n.$$

Each asset i has an *expected return* $r_i \in \mathbb{R}$. Moreover, for each pair of assets (i, j), we are given a *return covariance* $\sigma_{ij} \in \mathbb{R}$. The original problem formulation by Markowitz [34] includes the objectives of maximizing the expected portfolio return

$$\mu(\omega) := \sum_{i=1}^{n} \omega_i r_i$$

and minimizing the portfolio volatility

$$\sigma^2(\omega) := \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j \sigma_{ij} \ .$$

In general, the two objectives of maximizing return and minimizing volatility can not be fulfilled at the same time. Thus, several problem variants exist, which we summarize in the following. In the first variant, we aim at maximizing the return with an upper bound $\nu > 0$ on the volatility. This can be formulated by the quadratic program

$$
\textbf{MaxRet}: \quad \max_{\omega} \quad \mu(\omega)
$$
$$
\text{s.t.} \quad \sigma^2(\omega) \leq \nu
$$
$$
\sum_{i=1}^{n} \omega_i = 1
$$
$$
0 \leq \omega_i \leq u_i \quad \forall i \in \{1, \ldots, n\} \, .
$$

The second variant minimizes the volatility while ensuring a minimum return $\epsilon \in \mathbb{R}$. This is formalized by the quadratic program

$$
\textbf{MinVola}: \quad \min_{\omega} \quad \sigma^2(\omega)
$$
$$
\text{s.t.} \quad \mu(\omega) \geq \epsilon \tag{1}
$$
$$
\sum_{i=1}^{n} \omega_i = 1 \tag{2}
$$
$$
0 \leq \omega_i \leq u_i \quad \forall i \in \{1, \ldots, n\} \, .
$$

Finally, the third variant combines both objectives by a fixed factor $\lambda > 0$, which depends on the risk aversion of the investor. This gives rise to the quadratic program

$$
\textbf{MultiObj}: \quad \min_{\omega} \quad \sigma^2(\omega) - \lambda\mu(\omega)
$$
$$
\text{s.t.} \quad \sum_{i=1}^{n} \omega_i = 1
$$
$$
0 \leq \omega_i \leq u_i \quad \forall i \in \{1, \ldots, n\} \, .
$$

The mathematical programs MaxRet, MinVola and MultiObj all belong to the class of convex quadratically constrained quadratic programs, which can be solved in polynomial time by interior point methods [35]. The convexity is due to the fact that $(\sigma_{ij}) \in \mathbb{R}^{n \times n}$ is a sample covariance and thus positive semidefinite by definition (6) in Appendix A. Although being efficiently solvable in theory, portfolio optimization is often tackled via heuristics in practice.

Testset generation. In order to generate a realistic set of instances, we consider the daily prices of $1,978$ assets in the Nasdaq stock exchange [36] during the years 2020 to 2023, available in [37]. From the daily prices, we calculate the expected returns and the return covariances. For details on the calculation of returns and covariances from price data, we refer to Appendix A. To generate an instance with a given number n of assets, we randomly draw n assets from the data base.

We generate 10 instances for each problem variant MaxRet, MinVola and MultiObj for each size $n \in \{3, 5, 7, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000\}$. We still need to define values for the maximum portfolio volatility ϵ in MaxRet, the minimum portfolio return ν in MinVola and the risk aversion λ in MultiObj. As we strive towards an above-average performing portfolio, in Model MaxRet, we choose the maximum portfolio volatility ϵ as the 70 % quantile of covariances of randomly generated portfolios. Similarly, in Model MinVola, we choose the minimum portfolio return ν as the 70 % quantile of all asset returns. Moreover, in order to account for the different magnitudes of μ and σ^2, we set the risk aversion factor λ of Model MultiObj to the average of fraction $\frac{\sigma^2}{\mu}$ of randomly generated portfolios. Finally, we set the asset limits to $u_i = \max(\frac{1}{10}, \frac{3}{n})$ for all $i \in \{1, ..., n\}$, as for a larger number of assets we would like to have a more diversified portfolio.

In principle, some generated test instances of MinVola and MaxRet could be infeasible. However, our computational experiments revealed that all considered test instances are indeed feasible.

3 Classical Solution Time for Different Variants

In this section, we study the classical computational difficulty of the three problem variants MaxRet, MinVola and MultiObj. Our goal is to identify the hardest variant for classical solvers which thus bears
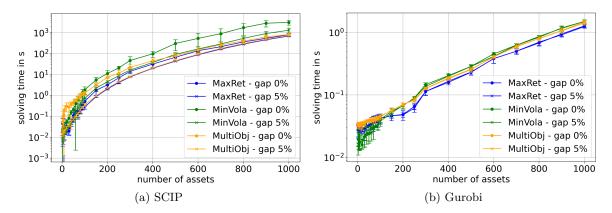
Figure 1: Average solver runtime for different problem variants. We compare SCIP (a) and Gurobi (b) with relative optimality gaps of 0 % and 5 %. Data points are averages over 10 instances and error bars show the empirical standard deviation.

the largest potential for a quantum advantage. To this end, we solve all instances of the test set described in Section 2 with classical MIP solvers to proven optimality and compare the overall runtime. For each instance, we solve the three problem variants with the open-source solver SCIP [38] and the commercial solver Gurobi [39]. Moreover, we impose a time limit of $3,600$ s and relative optimality gaps of 0 % and 5 %. Here, the relative optimality gap, short gap, is defined as

$$g := \frac{|b - c|}{|c|}$$

where b is the best bound on the objective and c is the objective value of the best available solution.

The average solving times are visualized in Figure 1. First, we observe that Gurobi solves the problems considerably faster than the open-source solver SCIP, with a speed-up factor of over $1,000$ for large problem instances with $n = 1,000$ assets. Second, the influence of the duality gap on the solving time is larger for SCIP than for Gurobi. Third, the Model MinVola takes the longest time to be solved in each of the four solver-gap configurations. Thus, MinVola is the most promising variant for a potential quantum speedup. Consequently, in the following we focus on the MinVola variant that minimizes volatility while still maintaining a certain portfolio return level.

4 QUBO Transformation

Existing implementations of both quantum annealing and QAOA require the input problem to be modeled as QUBO, that is, problems of the form

$$\min_{x \in \{0,1\}^n} x^t Q x$$

where $n \in \mathbb{N}$ and $Q \in \mathbb{R}^{n \times n}$ is a real-valued matrix. In this section, we derive a QUBO model of the MinVola variant. To this end, we define the penalty-based objective function

$$f(\omega) := \sigma^2 + \phi \left(\mu(\omega) - \epsilon \right)^2 + \psi \left(\sum_{i=1}^{n} \omega_i - 1 \right)^2, \tag{3}$$

where $\omega \in [0,1]^n$ are the asset weights and $\phi, \psi \in \mathbb{R}_{\geq 0}$ are penalty factors that penalize violations of the return constraint (1) and the normalization constraint (2), respectively. From a theoretical point of view, finding optimal penalty factors is itself an NP-hard problem [27]. Consequently, we employ a heuristic penalty approach in this work, which we detail in section 6.

In order to derive a valid QUBO model, the continuous variables $\omega_i \in [0,1]$ in (3) have to be transformed to binaries. As proposed in [13, 17], we write the portfolio weights $\omega_i \in [0,1]$ as a linear combination of binary variables $\omega_{ij} \in \{0,1\}$, $j \in \{1, \ldots, d+1\}$, via

$$\omega_i = u_i \cdot \left(\frac{1}{2^d} \omega_{i,d+1} + \sum_{j=1}^{d} \frac{1}{2^j} \omega_{i,j} \right) \quad \forall i \in 1, \ldots, n. \tag{4}$$

Here, $d \in \mathbb{N}$ controls the precision of discretization. An advantage of this discretization is the implicit incorporation of the upper bounds u_i on the asset weights. Inserting (4) in (3) yields the QUBO model for MinVola.

5 Methods

In this section, we introduce the solution methods we compare in our benchmark. We further explain how associated parameters are chosen.

MIP solver. In this benchmark, we chose Gurobi [39] as the MIP solver. Since MIP solvers can handle continuous variables, we solve the original, continuous formulation of MinVola. We use the MIP solver to calculate a provably optimal solution.

Quantum annealing. Quantum annealing is a heuristic algorithm that runs on analog quantum computers [40, 41]. The concept of quantum annealing is based on the adiabatic theorem of quantum mechanics [42]. Quantum annealing prepares a quantum mechanical system in the ground state of some initial Hamiltonian H_0. Then, it evolves the system according to

$$H(t) = A(t)H_0 + B(t)H_1 \, .$$

Here, H_1 is the problem Hamiltonian whose ground sate encodes the solution to some optimization problem. $A : [0, \tau] \to [0, 1]$ is a chosen such that $A(0) = 1$ and $A(\tau) = 0$, where $\tau > 0$ is called *annealing time*. Similarly, $B : [0, \tau] \to [0, 1]$ is chosen such that $B(0) = 0$ to $B(\tau) = 1$. Now, the adiabatic theorem states that the system is in the ground state of H_1 at $t = \tau$ for $\tau \gg 1$ if the energy gap between the ground state and the first excited state is non-zero for all $t \in [0, \tau]$ and if $\partial_t H(t)$ is finite for all $t \in [0, \tau]$. In our computational experiments, we use a D-Wave Advantage 2 processor which has up to 4,600 physical qubits [43]. However, the limited device connectivity requires an embedding of each QUBO variable into possibly multiple physical qubits [44, 45]. This embedding overhead increases with the density of the QUBO problem. All considered QUBO instance have a density of 100%. For embedding, we therefore use the `DWaveCliqueSampler` [46] which employs pre-defined embeddings for fully connected QUBO problems and is thus more suited for dense problems than constructive embedding algorithms like the default `MinorMiner` [47]. Also, we test annealing times of 1 μs, 5 μs, 20 μs and 50 μs. The *annealing schedule*, which defines $A(t)$ and $B(t)$, is kept as default. However, we vary the *chain strength* parameter which can prevent the breaking of a chain of connected qubits representing a single QUBO variable.

QAOA. The quantum approximate optimization algorithm is a quantum-classical hybrid algorithm, originally proposed by Farhi et al. [10]. Like quantum annealing, QAOA is conceptually based on the adiabatic theorem of quantum mechanics [42]. However, in contrast to quantum annealing, QAOA is designed for gate-based, universal quantum computers. QAOA depends on real-valued parameters $\gamma = (\gamma_1, ..., \gamma_p)$ and $\beta = (\beta_1, ..., \beta_p)$. The hyper-parameter $p \in \mathbb{N}$ controls the complexity of the algorithm. QAOA starts with the uniform superposition quantum state $|+\rangle^n$ and evolves it according to

$$|\psi(\beta, \gamma)\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \, ... \, e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} |+\rangle^n \, . \tag{5}$$

Here H_C is the so-called *problem Hamiltonian*, defined by

$$H_C |x\rangle = C(x) |x\rangle \quad \forall x \in \{0, 1\}^n$$

where $C(x)$ is the QUBO cost function. H_M is called the *mixing Hamiltonian* and is defined by

$$H_M = \sum_{i=1}^{n} X_i$$

where X_i is the Pauli X gate acting on the i-th qubit. For given parameters β and γ, solutions to the QUBO problem are sampled from the quantum circuit implementing (5). In this work, we employ three different methods to compute the parameters β and γ. First, for $p = 1$, we perform a grid search by analytically calculating the expectation value of the resulting QAOA circuit via the formula developed in [48]. We choose the parameter values that result in the minimum expectation. Second, we use the linear ramp QAOA (LR-QAOA) protocol introduced in [49]. Accordingly, we set $\beta_i = (1 - \frac{i}{p})\Delta_\beta$ and

$\gamma_i = (\frac{i+1}{p})\Delta_\gamma$ for $i \in \{1, \ldots, p\}$. In our experiments, we choose $\Delta_\beta = 0.3$ and $\Delta_\gamma = 0.6$ as proposed in [49]. Finally, we optimize the parameters with the local optimizer COBYLA [50] using a noiseless quantum simulator [51]. Here, we initialize the parameters with the LR-QAOA-formula.

After selecting parameters, we execute QAOA on the gate-based quantum computer *ibm_strasbourg* [52] and on an ideal quantum simulator [51]. Exponentially increasing memory requirements limit the classical simulation of QAOA to roughly 30 variables. Finally, we note that our method of calculating parameters has no optimality guarantee and that better QAOA parameters might exist.

Steepest Descent. The steepest descent local search tries to improve the current solution by investigating its neighborhood. Here, the neighborhood is defined as all solutions having a Hamming distance of one to the current solution. If the best solution in the neighborhood improves the objective value, it is accepted as the new current solution. Otherwise, the algorithm terminates. The main advantage of the steepest descent search is its fast runtime. A disadvantage is its inability to escape local minima. In our benchmark study, we employ the open-source implementation available in [53].

Simulated Annealing. Simulated annealing is a popular classical heuristic. The name of this algorithm is based on its similarity to the process of annealing in metallurgy, which aims to harden a metal by controlled heating and cooling [54]. In its general form, simulated annealing starts with a random solution. Then, simulated annealing iteratively considers a random solution in the neighborhood of the current solutions. If this solution improves upon the current objective value, it is accepted as the new current solution. Otherwise, it is still accepted as the new current solution with a probability decreasing with the number of iterations. By this methodology, the algorithm explores a large fraction of the solution space at the start, while at the end, objective-improving solutions are favored. The algorithm returns the solution that with the best objective value. Several parameters can be modified, for example, the definition of the neighborhood function and the acceptance probability function. In our computational study, we use an open-source implementation of simulated annealing [55].

Tabu Search. Tabu search is a classical heuristic conceptually similar to steepest descent. Tabu search starts with a random solution, then explores the neighborhood of this solution and chooses the best solution in this neighborhood while putting the previous solution on a tabu list. Solutions on the tabu list are excluded from any neighborhood. The tabu list has a limited length and solutions are removed in a first-in-first-out manner. The tabu list allows the algorithm to escape local minima. Termination criteria can be runtime, number of steps or objective value. Adjustable parameters are the selection of the neighborhood function, the termination criterion and the length of the tabu list. In our computational study we use an open-source implementation of tabu search [56].

Problem-specific heuristic. We also developed a problem-specific heuristic for the MinVola version of the portfolio optimization problem. For an instance with n assets, it creates up to n feasible solutions. The heuristic starts with an empty portfolio and then increases the asset weight ω_i of some asset $i \in \{1, \ldots, n\}$ by a fixed amount of $\delta > 0$. Then, it iteratively adds weight δ to the asset that reduces the portfolio variance the most while still satisfying constraint (1). The heuristic stops when the normalization constraint (2) is satisfied. For a more detailed description of the problem specific heuristic, we refer to appendix B.

6 Computational Experiments

In this section, we benchmark the solution methods discussed in Section 5 on the portfolio optimization problem test set introduced in Section 2. The goal of our computational experiments is to quantify the potential of currently available quantum computers for portfolio optimization. Ours study follows the general rules on good-practice for benchmarks from [57], which are relevance, reproducibility, fairness, verifiability and usability. Our code and data are publicly available at [58].

Benchmark Procedure. We compare the solution methods introduced in Section 5, that is, QAOA, quantum annealing, simulated annealing, steepest descent, tabu search and the problem-specific heuristic on the MinVola problem variant.

In the QUBO model (3), we set the penalty factors $\phi = \psi = 1000$ for violations of the return constraint (1) and the normalization constraint (2), respectively. We determine these factors by analyzing the magnitude

of the objective terms. Exemplary, for 500 assets, the portfolio returns in our benchmark set fluctuate in a range of $[-0.1, 0]$ and the portfolio volatilities fluctuate in the range of $[0.05, 1]$. For smaller problem instances with 10 assets, the returns vary between $[-0.3, 0.5]$ and the volatilities between $[0.06, 3]$. A small absolute constraint violation of 0.01 in the constraints (1) or (2) will lead to a penalization of $1000 \cdot 0.01^2 = 0.1$. Thus, the penalization is in the same order of magnitude as the objective value. Additionally, we choose the discretization coarseness in (4) as $d = 3$. This leads to $d + 1 = 4$ times as many variables in the discretized model as in the continuous model. Consequently, for an instance with n assets, the resulting QUBO has $4n$ variables. The value of $d = 3$ results from a trade-off between QUBO size and discretization error.

To ensure fairness among the methods, we enforce a time limit of 60 seconds on the solution process. For the grid-search QAOA, we split up the 60 seconds into 30 seconds of training and 30 seconds of sampling with the best found parameters. We configure the size of the grid such that its execution time takes roughly 30 seconds. For the QAOA with optimized parameters, we spent a maximum of 52 seconds on training while the rest is used to exclusively sample with the best found parameters. This ensures that we have at least a minimum amount of time for sampling with high-quality parameters. After 60 seconds, we return the best found solution. This resembles a practical setting where one is usually interested in the best solution rather than statistical performance measures like expectation values or quantiles. Of course, the sampling results found during the training of QAOA are also considered in the final result. As our primary quality metric, we consider the *approximation ratio* which is defined as

$$\Theta := \frac{f_m}{f_{\text{opt}}} \geq 1.$$

Here f_m is the objective value returned by the method m, and f_{opt} is the value of the optimal solution, which we calculate by a MIP solver. We have $f_m \geq f_{\text{opt}} \geq 0$ since MinVola is a minimization problem and the portfolio volatility is always non-negative. Thus, a value of Θ close to 1 indicates near-optimal solutions.

In addition, we measure the *feasibility percentage*, which we define as the fraction of the 10 test instances per problem size for which the considered method returns at least a single feasible solution.

Finally, the *number of samples* measures how many samples are generated by the respective algorithm within the time limit.

Benchmark Results. Figure 2 summarizes our results. We first discuss the results of quantum annealing, shown in Figures 2a, 2b and 2c. We observe that quantum annealing with the default calculation of the chain strength (red and pink lines) yields the worst approximation ratio (Figure 2a) and the lowest feasibility percentage (Figure 2b). In particular, even random sampling has a lower average approximation ratio on the same number of samples (yellow line in Figure 2a) while finding more feasible solutions (yellow line in Figure 2b). After testing different chain strength parameters for different instance sizes, we propose the following improved chain strength values. For problems of 5, 7, 10 and 15 assets we set $cs = 3$, for 20 assets we set $cs = 4$ and for 25 assets we set $cs = 5$. The results of quantum annealing with improved chain strength are shown in the black and gray lines in Figures 2a, 2b and 2c. We observe that quantum annealing with improved chain strength is roughly en par with random sampling in terms of both feasibility percentage and approximation ratio. Nevertheless, we observe in Figures 2a and 2b that quantum annealing performs significantly worse than the problem-specific heuristic. We conjecture that the high density of the QUBO problem is a reason for the suboptimal performance of quantum annealing.. Embedding the required all-to-all connectivity leads to long chains of qubits for each variable. Emphasizing the large embedding overhead, we note that the largest problem size that can still be embedded on the quantum annealing processor with 4597 physical qubits has 25 assets, which corresponds to 100 QUBO variables. Moreover, varying the annealing time between $20\mu s$ and $50\mu s$ did not significantly improve the results, compare pink versus red and grey versus black in Figures 2a and 2b. We remark that we have conducted additional experiments with annealing times of $1\mu s$ and $5\mu s$. Those results did not differ significantly from the results with annealing times of $20\mu s$ and $50\mu s$. Finally, in Figure 2c, we observe that the number of samples generated in 60 s is independent of the problem size. From this, we conclude that the runtime of quantum annealing does not significantly increase with problem size, which might lead to an advantage of future annealing hardware for very large instances.

Next, we turn to the results for QAOA, shown in Figures 2d, 2e, 2f. Similar to quantum annealing, QAOA often falls short on finding feasible solutions, which results in decaying feasibility percentages in Figure 2e. Even when QAOA returns feasible solutions, their approximation ratio quickly turns sub-optimal when increasing the problem size. We observe approximation values of ≥ 2 for problem with 20 or more assets in

Figure 2d. In general, QAOA does not perform significantly better than random sampling with respect to feasibility and approximation ratio, Compared to the problem-specific heuristic, all QAOA configurations deliver lower feasibility percentages and worse approximation ratios. The QAOA configuration with the best results is the 1-layer linear-ramp (red lines) which returns a lower approximation ratio on a higher feasibility percentage on a similar number of samples than the remaining QAOA configuration. Moreover, we observe increasing the number of layers worsens the performance although, in theory, the performance improves with the number of layers. Other experiments with 2 and 4 layers affirm this observation. We do not show their results here for clarity. We attribute this paradox mainly to two reasons. First, more layers increase the circuit execution time. Consequently, less shots can be executed in the same time, which can be seen in Figure 2f. Second, the effect of noise increases with the circuit size. A reason for the decaying performance of QAOA with larger problem instances is the rapidly increasing number of swap gates that are required to transpile the QAOA circuit with all-to-all-connectivity to the quantum computer with limited connectivity. For example, for 20 assets, the circuit depth of 1-layer QAOA increases from 161 to $\approx 11,000$ during transpilation. We did not include our results for the classically simulated QAOA. The reason is that the largest instance which can be simulated has 7 assets. Finally, we remark that we did not run QAOA experiments in which we optimize the parameters on the hardware since the communication times are prohibitively large.

Lastly, we compare the results for classical heuristics in Figures 2g, 2h and 2i. All open-source implementations (steepest descent, simulated annealing and tabu search) are executed with default settings. In general, all heuristics are able to find feasible solutions for nearly all problem instances, see Figure 2h. Only tabu search does not return feasible solutions for all instances with 800 assets. On the other hand, random sampling failed to return any feasible solutions starting from 40 assets. The approximation ratios of simulated annealing, steepest descent and tabu search exceed the value of 2 starting at 50 assets and the value of 10 at 250 or more assets. The problem-specific heuristic performs best with respect to the average approximation ratio. The approximation ratios of the other heuristics are comparable despite differing significantly in the numbers of samples (Figure 2i). We remark that all heuristics rely on objective evaluation which can take a significant amount of time on large instances. This explains the decreasing number of samples for larger problem instances. Exemplary, for 500 assets, the problem-specific heuristic is only able to return roughly 100 out of 500 possible solutions in 60 s. Thus, quantum methods might be beneficial for even higher asset numbers since there is no objective evaluation involved in the quantum sampling process. Nevertheless, the performance of the problem-specific heuristic sets the bar high for a possible quantum advantage
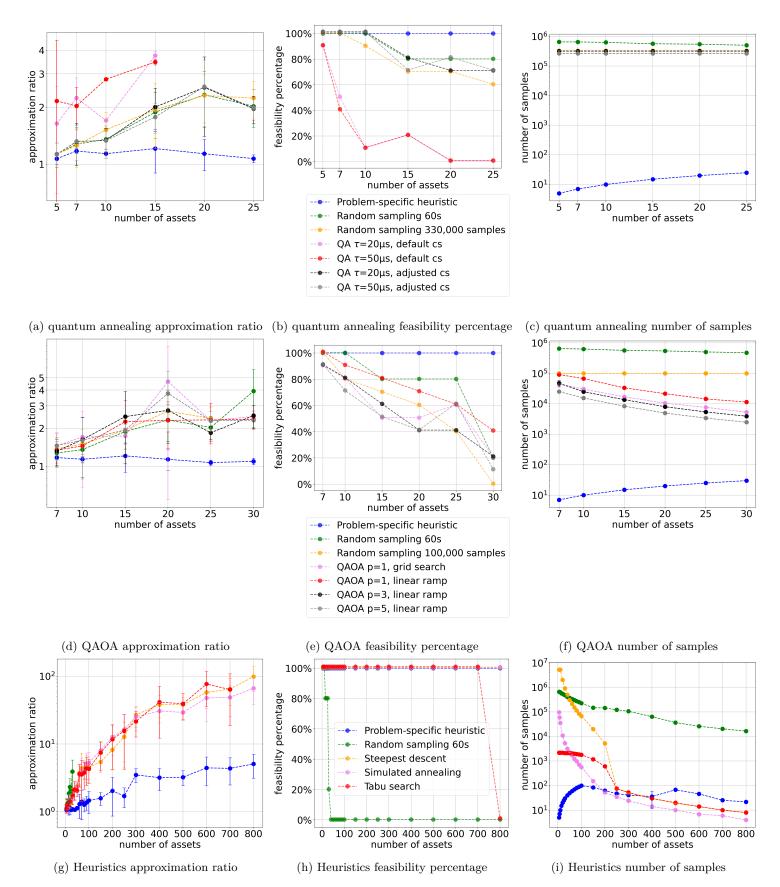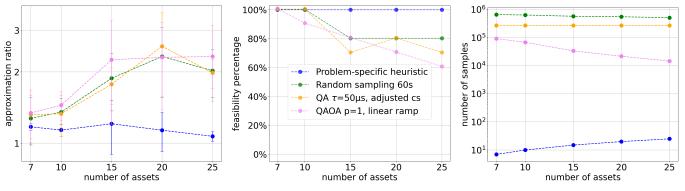
(a) quantum annealing approximation ratio   (b) quantum annealing feasibility percentage   (c) quantum annealing number of samples

(d) QAOA approximation ratio   (e) QAOA feasibility percentage   (f) QAOA number of samples

(g) Heuristics approximation ratio   (h) Heuristics feasibility percentage   (i) Heuristics number of samples

Figure 2: Summarized computational results. We report the approximation ratio, feasibility percentage and number of samples for quantum annealing (a - c), QAOA (d - f) and classical heuristics (g - i). For quantum annealing and QAOA, we compare against the problem-specific heuristic and random sampling as baselines. Data points are averages over 10 instances and error bars show the empirical standard deviation. Missing data points in the approximation ratio are due to the absence of feasible solutions. For each row, we display a single legend which holds for all three subfigures.

(a) Best methods: approximation ratio (b) Best methods: feasibility percentage (c) Best methods: number of shots

Figure 3: Comparison of the best performing configurations from our computational study. We report the approximation ratio (a), the feasibility percentage (b) and the number of samples generated during the given time limit of 60 seconds (c). Data points are averages over 10 instances and error bars show the empirical standard deviation. The legend holds for all three subfigures.

In Figures 3a, 3b and 3c, we compare the most promising approaches from quantum annealing, QAOA and classical heuristics in order to rank them. In particular, we consider quantum annealing with 50 μs annealing time and adjusted chain strength, the single-layer QAOA with linear ramp parameters and the problem-specific heuristic.

We observe that the problem-specific heuristic has a larger feasibility percentage and a better approximation ratio than the best configurations of both QAOA and quantum annealing. Furthermore, both QAOA and quantum annealing perform roughly as good as random sampling. Notably, random sampling generates significantly more solutions in 60 s (Figure 3c). Finally, we conclude that quantum annealing slightly outperforms QAOA since quantum annealing generates more samples, finds more feasible solutions and those solutions are also of slightly higher quality. The difference in the number of generated samples would get larger for increasing problem size since the QAOA circuit length and thus its execution time grows with the number of variables whereas the annealing time stays constant.

7 Conclusion

In this work, we conducted an extensive benchmark of quantum computing for portfolio optimization. Recently, several works suggested portfolio optimization as a suitable candidate for a possible quantum advantage. Our experiments, however, highlight the challenges in achieving such an advantage in practice. We consider a variant of portfolio optimization which we have shown to be particularly difficult for classical optimizers. We then compared both classical and quantum methods on 250 problem instances from real-world stock data. In our benchmark, we imposed a time limit of 60 seconds to ensure fairness among the methods. Our main conclusion is that classical heuristics like simulated annealing, steepest descent, tabu search and a problem-specific heuristic clearly outperform QAOA and quantum annealing regarding solution feasibility and quality. Regarding classical exact optimizers, Gurobi and SCIP differ significantly in their solution times. Here, Gurobi is more than 1000 times faster for large problem instances, solving problems with 1000 assets in the order of seconds. Comparing quantum methods, we observed that quantum annealing slightly outperformed the gate-based QAOA in terms of feasibility and quality. However, this superiority of quantum annealing only appeared after parameter fine-tuning. While being outperformed by classical heuristics, QAOA and quantum annealing also did not clearly differ from random sampling within the time limitation of 60 seconds. Here, we emphasize that we observed a clear difference in the distribution of objective values when comparing QAOA and quantum annealing to random sampling. However, in this work we considered only the best solution instead of statistical measures of the solution distribution. We conjecture the all-to-all connectivity of the problem to be the main reason for the poor performance of QAOA and quantum annealing. Dense problems generate a large qubit overhead during embedding and a large gate overhead during transpilation. Finally, we remark that the considered problem variant is the most simple formulation of portfolio optimization. For more complicated problem variants with additional constraints and variables, further experiments are required to study the performance of quantum methods.

Acknowledgements

Appendix

A   Generating stock data

For our benchmark, we create portfolio problem instances from real world stock data. We use the Python package yfinance [59] to retrieve historical NASDAQ closing prices $p_{i,t}$ of asset i at time t. From the price data, we calculate the *daily asset returns*

$$rd_{i,t} = \frac{p_{i,t}}{p_{i,t-1}} - 1 \quad \forall i \in N, \ t \in T$$

and the *average daily asset returns*

$$rav_i = \frac{1}{|T|} \sum_{t \in T} rd_{it} \quad \forall i \in N.$$

For the portfolio optimization problem, we work with *annualized asset returns*, which are defined by

$$r_i := \left(\prod_{t \in T} (1 + rd_{it}) \right)^{\frac{252}{|T|}} \quad \forall i \in N.$$

The value of 252 is the average yearly amount of business days at the stock exchange. The estimated *annualized covariances* for all asset combinations of asset (i, j) are calculated by

$$\sigma_{ij} := \frac{252}{|T|} \sum_{t \in T} (rd_{i,t} - rav_i)(rd_{j,t} - rav_j) \quad \forall i, j \in N. \tag{6}$$

Now, the annualized asset returns r_i and the annualized asset covariances σ_{ij} are used to create instances of the portfolio optimization problem.

B   Pseudocode Problem-specific Heuristic

The Problem-specific heuristic for the MinVola problem formulation with $n \in \mathbb{N}$ possible assets generates up to n feasible solutions.

Algorithm 1 MinVola - Problem-specific Heuristic

1: Let $\mu_i \in \mathbb{R} \leftarrow$ expected return for asset $i \in \{1, \ldots, n\}$
2: Let $u_i \in (0, 1] \leftarrow$ upper bound of asset weight for asset $i \in \{1, \ldots, n\}$
3: Let $\sigma_{ij} \in \mathbb{R} \leftarrow$ covariance of returns of assets $i, j \in \{1, \ldots, n\}$
4: Let $\nu \in \mathbb{R} \leftarrow$ minimum required portfolio return
5: Let $\delta \in (0, 1] \leftarrow$ constant weight parameter that gets added to the samples
6: Initialize FeasSols $\leftarrow$ empty list of feasible solutions
7: **for** $i \leftarrow 1$ **to** n **do**
8:     Initialize $x \leftarrow [0, \ldots, 0]$ empty asset weight vector of length n
9:     $x_i \leftarrow \delta$ add weight to asset i
10:     **while** $\sum_{k=1}^{n} x_k < 1$ **do**
11:         $x \leftarrow$ AddNewAssetWeightSteepestDesc$(x, \mu, ub, \sigma, \delta, \epsilon)$
12:     **end while**
13:     Add x to FeasSols
14: **end for**
15: **return** FeasSols

16: **function** AddNewAssetWeightSteepestDesc$(x, \mu, ub, \sigma, \delta, \epsilon)$
17:     Let big_M $\leftarrow 1000$
18:     Let small_M $\leftarrow -1000$
19:     Initialize $Vols \leftarrow [\,]$ volatilities
20:     Initialize $Rets \leftarrow [\,]$ returns
21:     **for** $j \leftarrow 1$ **to** n **do**
22:         Let $x' \leftarrow$ copy of x
23:         $x'_j \leftarrow x'_j + \delta$
24:         **if** $x'_j \leq ub_j$ **then**
25:             $r \leftarrow \dfrac{\sum_{k=1}^{n} x'_k \mu_k}{\sum_{k=1}^{n} x'_k}$ calculate normalized return
26:             Append r to $Rets$
27:             **if** $r \geq \nu$ **then**
28:                 $\nu \leftarrow \sum_{k=1}^{n} \sum_{l=1}^{n} x'_k \sigma_{kl} x'_l$ calculate return volatility
29:                 Append ν to $Vols$
30:             **else**
31:                 Append big_M to $Vols$
32:             **end if**
33:         **else**
34:             Append small_M to $Rets$
35:             Append big_M to $Vols$
36:         **end if**
37:     **end for**
38:     **if** $\min(Vols) =$ big_M **then**
39:         $j^* \leftarrow \arg\max Rets$
40:     **else**
41:         $j^* \leftarrow \arg\min Vols$
42:     **end if**
43:     $x_{j^*} \leftarrow x_{j^*} + \delta$
44:     **return** x
45: **end function**

References

[1] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithms and Combinatorics*. Springer, Berlin, Heidelberg, 2018. URL: https://link.springer.com/10.1007/978-3-662-56039-6.

[2] Thorsten Koch, Timo Berthold, Jaap Pedersen, and Charlie Vanaret. Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, 10:100031, 2022. doi:10.1016/j.ejco.2022.100031.

[3] Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer, Berlin, Heidelberg, 2010. URL: http://link.springer.com/10.1007/978-3-540-68279-0.

[4] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 13(3):443–490, September 2021. doi:10.1007/s12532-020-00194-3.

[5] Amira Abbas, Andris Ambainis, Brandon Augustino, Andreas Bärtschi, Harry Buhrman, Carleton Coffrin, Giorgio Cortiana, Vedran Dunjko, Daniel J. Egger, Bruce G. Elmegreen, Nicola Franco, Filippo Fratini, Bryce Fuller, Julien Gacon, Constantin Gonciulea, Sander Gribling, Swati Gupta, Stuart Hadfield, Raoul Heese, Gerhard Kircher, Thomas Kleinert, Thorsten Koch, Georgios Korpas, Steve Lenk, Jakub Marecek, Vanio Markov, Guglielmo Mazzola, Stefano Mensa, Naeimeh Mohseni, Giacomo Nannicini, Corey O'Meara, Elena Peña Tapia, Sebastian Pokutta, Manuel Proissl, Patrick Rebentrost, Emre Sahin, Benjamin C. B. Symons, Sabine Tornow, Víctor Valls, Stefan Woerner, Mira L. Wolf-Bauwens, Jon Yard, Sheir Yarkoni, Dirk Zechiel, Sergiy Zhuk, and Christa Zoufal. Challenges and opportunities in quantum optimization. *Nature Reviews Physics*, 6(12):718–735, December 2024. Publisher: Nature Publishing Group. URL: https://www.nature.com/articles/s42254-024-00770-9.

[6] Sabrina Ammann, Maximilian Hess, Debora Ramacciotti, Sándor P. Fekete, Paulina L. A. Goedicke, David Gross, Andreea Lefterovici, Tobias J. Osborne, Michael Perk, Antonio Rotundo, S. E. Skelton, Sebastian Stiller, and Timo de Wolff. Realistic Runtime Analysis for Quantum Simplex Computation, November 2023. arXiv:2311.09995 [quant-ph]. URL: http://arxiv.org/abs/2311.09995.

[7] Alexey Bochkarev, Raoul Heese, Sven Jäger, Philine Schiewe, and Anita Schöbel. Quantum Computing for Discrete Optimization: A Highlight of Three Technologies, September 2024. arXiv:2409.01373 [math]. URL: http://arxiv.org/abs/2409.01373.

[8] Tameem Albash and Daniel A. Lidar. Demonstration of a Scaling Advantage for a Quantum Annealer over Simulated Annealing. *Phys. Rev. X*, 8(3):031016, July 2018. Publisher: American Physical Society. doi:10.1103/PhysRevX.8.031016.

[9] Catherine C. McGeoch and Pau Farré. Milestones on the Quantum Utility Highway: Quantum Annealing Case Study. *ACM Transactions on Quantum Computing*, 5(1), December 2023. Place: New York, NY, USA Publisher: Association for Computing Machinery. doi:10.1145/3625307.

[10] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm, November 2014. arXiv:1411.4028 [quant-ph]. URL: http://arxiv.org/abs/1411.4028.

[11] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Daniel Eppens, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Alan Ho, Sabrina Hong, Trent Huang, L. B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Martin Leib, Orion Martin, John M.

Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Charles Neill, Florian Neukart, Murphy Yuezhen Niu, Thomas E. O'Brien, Bryan O'Gorman, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Andrea Skolik, Vadim Smelyanskiy, Doug Strain, Michael Streif, Marco Szalay, Amit Vainsencher, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Leo Zhou, Hartmut Neven, Dave Bacon, Erik Lucero, Edward Farhi, and Ryan Babbush. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, February 2021. Publisher: Springer Science and Business Media LLC. doi:10.1038/s41567-020-01105-y.

[12] Thorsten Koch, David E. Bernal Neira, Ying Chen, Giorgio Cortiana, Daniel J. Egger, Raoul Heese, Narendra N. Hegade, Alejandro Gomez Cadavid, Rhea Huang, Toshinari Itoko, Thomas Kleinert, Pedro Maciel Xavier, Naeimeh Mohseni, Jhon A. Montanez-Barrera, Koji Nakano, Giacomo Nannicini, Corey O'Meara, Justin Pauckert, Manuel Proissl, Anurag Ramesh, Maximilian Schicker, Noriaki Shimada, Mitsuharu Takeori, Victor Valls, David Van Bulck, Stefan Woerner, and Christa Zoufal. Quantum Optimization Benchmark Library – The Intractable Decathlon, April 2025. arXiv:2504.03832 [quant-ph]. URL: http://arxiv.org/abs/2504.03832.

[13] Wolfgang Sakuler, Johannes M. Oberreuter, Riccardo Aiolfi, Luca Asproni, Branislav Roman, and Jürgen Schiefer. A real world test of Portfolio Optimization with Quantum Annealing, March 2023. arXiv:2303.12601 [quant-ph]. URL: http://arxiv.org/abs/2303.12601.

[14] Chen-Yu Liu and Hsi-Sheng Goan. Hybrid Gate-Based and Annealing Quantum Computing for Large-Size Ising Problems, August 2022. arXiv:2208.03283 [quant-ph]. URL: http://arxiv.org/abs/2208.03283.

[15] Atithi Acharya, Romina Yalovetzky, Pierre Minssen, Shouvanik Chakrabarti, Ruslan Shaydulin, Rudy Raymond, Yue Sun, Dylan Herman, Ruben S. Andrist, Grant Salton, Martin J. A. Schuetz, Helmut G. Katzgraber, and Marco Pistoia. Decomposition Pipeline for Large-Scale Portfolio Optimization with Applications to Near-Term Quantum Computing, November 2024. arXiv:2409.10301 [math]. URL: http://arxiv.org/abs/2409.10301.

[16] Frank Phillipson and Harshil Singh Bhatia. Portfolio Optimisation Using the D-Wave Quantum Annealer. In Maciej Paszynski, Dieter Kranzlmüller, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2021*, pages 45–59, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-77980-1_4.

[17] Samuel Palmer, Serkan Sahin, Rodrigo Hernandez, Samuel Mugel, and Roman Orus. Quantum Portfolio Optimization with Investment Bands and Target Volatility, August 2021. arXiv:2106.06735 [q-fin]. URL: http://arxiv.org/abs/2106.06735.

[18] Davide Venturelli and Alexei Kondratyev. Reverse Quantum Annealing Approach to Portfolio Optimization Problems. *Quantum Machine Intelligence*, 1(1-2):17–30, May 2019. arXiv:1810.08584 [quant-ph, q-fin]. URL: http://arxiv.org/abs/1810.08584.

[19] Sebastian Brandhofer, Daniel Braun, Vanessa Dehn, Gerhard Hellstern, Matthias Hüls, Yanjun Ji, Ilia Polian, Amandeep Singh Bhatia, and Thomas Wellens. Benchmarking the performance of portfolio optimization with QAOA. *Quantum Information Processing*, 22(1):25, December 2022. arXiv:2207.10555 [quant-ph]. URL: http://arxiv.org/abs/2207.10555.

[20] Mark Hodson, Brendan Ruck, Hugh Ong, David Garvin, and Stefan Dulman. Portfolio rebalancing experiments using the Quantum Alternating Operator Ansatz, November 2019. arXiv:1911.05296 [quant-ph]. URL: http://arxiv.org/abs/1911.05296.

[21] Zhijie Tang, Alex Lu Dou, and Arit Kumar Bishwas. Comparative analysis of diverse methodologies for portfolio optimization leveraging quantum annealing techniques, July 2024. arXiv:2403.02599 [quant-ph]. URL: http://arxiv.org/abs/2403.02599.

[22] Giuseppe Buonaiuto, Francesco Gargiulo, Giuseppe De Pietro, Massimo Esposito, and Marco Pota. Best practices for portfolio optimization by quantum computing, experimented on real quantum devices. *Scientific Reports*, 13(1):19434, November 2023. URL: https://www.nature.com/articles/s41598-023-45392-w.

[23] Francesco Cesarone, Andrea Scozzari, and Fabio Tardella. Efficient Algorithms for mean-variance portfolio optimization with Hard Real -World Constraints.

[24] Álvaro Rubio-García, Juan José García-Ripoll, and Diego Porras. Portfolio optimization with discrete simulated annealing, October 2022. arXiv:2210.00807 [cond-mat]. URL: http://arxiv.org/abs/2210.00807.

[25] T.-J. Chang, N. Meade, J.E. Beasley, and Y.M. Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27(13):1271–1302, November 2000. URL: https://linkinghub.elsevier.com/retrieve/pii/S030505489900074X.

[26] Cassidy K. Buhler and Hande Y. Benson. Efficient Solution of Portfolio Optimization Problems via Dimension Reduction and Sparsification, June 2023. arXiv:2306.12639 [q-fin]. URL: http://arxiv.org/abs/2306.12639.

[27] Edoardo Alessandroni, Sergi Ramos-Calderer, Ingo Roth, Emiliano Traversi, and Leandro Aolita. Alleviating the quantum Big-M problem. *npj Quantum Information*, 11(1):125, July 2025. URL: https://www.nature.com/articles/s41534-025-01067-0.

[28] Mahdi Moeini. A Continuous Optimization Approach for the Financial Portfolio Selection under Discrete Asset Choice Constraints, April 2014. arXiv:1404.3286 [cs]. URL: http://arxiv.org/abs/1404.3286.

[29] Francesco Cesarone, Andrea Scozzari, and Fabio Tardella. Portfolio selection problems in practice: a comparison between linear and quadratic optimization models. *Computational Management Science*, 12(3):345–370, July 2015. arXiv:1105.3594 [q-fin]. URL: http://arxiv.org/abs/1105.3594.

[30] Sarat Moka, Matias Quiroz, Vali Asimit, and Samuel Muller. A Scalable Gradient-Based Optimization Framework for Sparse Minimum-Variance Portfolio Selection, May 2025. arXiv:2505.10099 [stat]. URL: http://arxiv.org/abs/2505.10099.

[31] IBM ILOG CPLEX Optimization Studio, December 2022. URL: https://www.ibm.com/docs/en/icos/22.1.1?topic=cplex-meet.

[32] Vicente P. Soloviev, Antonio Márquez Romero, Josh Kirsopp, and Michal Krompiec. Scaling Portfolio Diversification with Quantum Circuit Cutting Techniques, June 2025. arXiv:2506.08947 [quant-ph]. URL: http://arxiv.org/abs/2506.08947.

[33] Zi Xuan Loke, Say Leng Goh, Graham Kendall, Salwani Abdullah, and Nasser R. Sabar. Portfolio Optimization Problem: A Taxonomic Review of Solution Methodologies. *IEEE Access*, 11:33100–33120, 2023. URL: https://ieeexplore.ieee.org/document/10087257/.

[34] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1952.tb01525.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x.

[35] Margaret Wright. The interior-point revolution in optimization: History, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56, September 2004. URL: https://www.ams.org/bull/2005-42-01/S0273-0979-04-01040-7/.

[36] NASDAQ Composite (^IXIC) Charts, Data & News. URL: https://finance.yahoo.com/quote/%5EIXIC/.

[37] Yahoo Finance - Stock Market Live, Quotes, Business & Finance News. URL: https://finance.yahoo.com/.

[38] Suresh Bolusani, Mathieu Besancon, Ksenia Bestuzheva, Antonia Chmiela, Joao Dionısio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, and Christopher Hojny. The SCIP Optimization Suite 9.0.

[39] The Leader in Decision Intelligence Technology - Gurobi Optimization. URL: https://www.gurobi.com/.

[40] Philipp Hauke, Helmut G. Katzgraber, Wolfgang Lechner, Hidetoshi Nishimori, and William D. Oliver. Perspectives of quantum annealing: Methods and implementations. *Reports on Progress in Physics*, 83(5):054401, May 2020. arXiv:1903.06559 [quant-ph]. URL: http://arxiv.org/abs/1903.06559.

[41] Tameem Albash. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), 2018. doi:10.1103/RevModPhys.90.015002.

[42] Albert Messiah. *Quantum Mechanics*. Elsevier, 1961. Google-Books-ID: VR93vUk8d_8C.

[43] The Advantage™ Quantum Computer | D-Wave. URL: https://www.dwavequantum.com/solutions-and-products/systems/.

[44] Vicky Choi. Minor-Embedding in Adiabatic Quantum Computation: I. The Parameter Setting Problem, April 2008. arXiv:0804.4884 [quant-ph]. URL: http://arxiv.org/abs/0804.4884.

[45] Vicky Choi. Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. *Quantum Information Processing*, 10(3):343–353, June 2011. arXiv:1001.3116 [quant-ph]. URL: http://arxiv.org/abs/1001.3116.

[46] Samplers — dwave-system 0.9.12 documentation. URL: https://dwave-systemdocs.readthedocs.io/en/latest/reference/samplers.html#dwavecliquesampler.

[47] Jun Cai, William G. Macready, and Aidan Roy. A practical heuristic for finding graph minors, June 2014. arXiv:1406.2741 [quant-ph]. URL: http://arxiv.org/abs/1406.2741.

[48] Asier Ozaeta, Wim Van Dam, and Peter L McMahon. Expectation values from the single-layer quantum approximate optimization algorithm on Ising problems. *Quantum Science and Technology*, 7(4):045036, October 2022. URL: https://iopscience.iop.org/article/10.1088/2058-9565/ac9013.

[49] J. A. Montanez-Barrera and Kristel Michielsen. Towards a universal QAOA protocol: Evidence of quantum advantage in solving combinatorial optimization problems, May 2024. arXiv:2405.09169 [quant-ph]. URL: http://arxiv.org/abs/2405.09169.

[50] minimize(method='COBYLA') — SciPy v1.15.3 Manual. URL: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-cobyla.html.

[51] AerSimulator - Qiskit Aer 0.16.1. URL: https://qiskit.github.io/qiskit-aer/stubs/qiskit_aer.AerSimulator.html.

[52] IBM Quantum. URL: https://quantum.ibm.com/.

[53] dwavesystems/dwave-greedy, September 2024. original-date: 2019-08-02T10:10:41Z. URL: https://github.com/dwavesystems/dwave-greedy.

[54] Simulated annealing, September 2025. Page Version ID: 1311714758. URL: https://en.wikipedia.org/w/index.php?title=Simulated_annealing&oldid=1311714758.

[55] dwave-samplers — Python documentation. URL: https://docs.dwavequantum.com/en/latest/ocean/api_ref_samplers/index.html.

[56] dwavesystems/dwave-tabu, September 2024. original-date: 2018-08-28T02:14:30Z. URL: https://github.com/dwavesystems/dwave-tabu.

[57] Arturo Acuaviva, David Aguirre, Rubén Peña, and Mikel Sanz. Benchmarking Quantum Computers: Towards a Standard Performance Evaluation Approach, July 2024. arXiv:2407.10941 [quant-ph]. URL: http://arxiv.org/abs/2407.10941.

[58] stopfereric/portfolio_opt_benchmark. URL: https://github.com/stopfereric/portfolio_opt_benchmark.

[59] yfinance: Download market data from Yahoo! Finance API. URL: https://github.com/ranaroussi/yfinance.