Project 2: Quantum for Portfolio Optimization

Czcibor Ciostek Weronika Golletz

Project Summary

1. We began by reviewing the mathematical formulation provided, focusing on binary decision variables, linear constraints, and the quadratic objective. We analyzed and understood the provided codebase and results from the GitHub repository. Rather than starting from scratch, we aimed to build upon this foundation to conduct further research and experimentation (Section 1).
2. We identified key opportunities for improving the Variational Quantum Eigensolver (VQE) optimization process, with a particular focus on:
   (a) The choice and structure of the ansatz (intro. in Section 2): For the 31-bond problem, we selected the TwoLocal and bfcd ansatzes with CVaR parameter $\alpha = 0.1$, repetition depth $r = 2$, and bilinear entanglement. This configuration achieved the most optimal solutions and fastest convergence among the tested parameter values, and was used in subsequent analyses.
   (b) The penalty parameter in constraint embedding (Section 2.1): We tested $\lambda \in \{1.1, 1.5, 2.0, 2.5\}$. The lowest penalty $\lambda = 1.1$ yielded the best convergence and optimization results for both ansatzes. However, constraint satisfaction was 80% for TwoLocal at $\lambda = 1.1$, while $\lambda = 1.5$ gave the highest satisfaction rate for bfcd (90%).
   (c) The circuit initialization strategy (Section 2.2): We compared the default initialization of all parameters to $\pi/3$ against random initialization. The $\pi/3$ initialization proved to be a reasonable choice: it gave the highest constraint satisfaction rates (90% for TwoLocal, 70% for bfcd), while random initialization produced mixed results—sometimes improving convergence, but often not.
3. To test scalability and performance, we developed a custom generator for producing problem instances of the same type as the original (binary quadratic problems with linear inequality constraints), but for varying numbers of bonds (Section 1.3). This allowed us to
   (a) Study scaling behavior across problem sizes (Section 3.1) – we analyzed problems with 15, 20, 25, 30 bonds, and found that while solution time increases predictably with size, the relative gap to the optimum also grows, with variance widening for larger instances.
   (b) Perform statistical evaluations across multiple runs and configurations (Section 3.2) – comparing 100 different 30-bond problems to 100 attempts on the same problem shows that certain problem instances are substantially harder, with some exhibiting objective values more than 50% worse than the optimum.
4. Motivated by recent literature, we also implemented a new ansatz – the Heat Exchange ansatz (Section 4).
5. Finally, we outlined future research directions to further improve solution quality and scalability. (Section 5).

1 Introduction

Complex optimization problems are one of the venues where quantum speedup promises major advantage over classical algorithms. While linear programs (LPs) with linear constraints have a well-defined mathematical structure and have well-understood classical solutions, changing the objective structure or domain of the variables drastically increases the difficulty of optimization problems.
Specifically, mixed-integer quadratic programs (MIQPs) are in general NP-hard:

$$\begin{array}{ll} \text{minimize} & x^T Q x + q^T x + r \\ \text{subject to} & x \in \mathcal{C} \\ & x \in \mathbf{Z}^n, \end{array} \tag{1}$$

where $x \in \mathbf{Z}^n$ is the optimization variable ($\mathbf{Z}^n$ is the set of n-dimensional vectors with integer-valued components), $Q \in \mathbf{S}_+^n$ (the set of $n \times n$ symmetric positive semidefinite matrices), $q \in \mathbf{R}^n$, and $r \in \mathbf{R}$ are problem data, and $\mathcal{C}$ is some convex set.
Here some of the variables are continuous, selecting amounts of divisible goods, money allocation or similar, while the rest of variables are of binary on/off selection kind, adjusting whether it is optimal to include a resource at all.
Ability to quickly solve large MIQPs (problems with thousands of variables and constraints are notoriously hard to solve using classical optimizers) are of high value in applications, including: resource allocation problems, index tracking and portfolio optimization.
Quantum computers are uniquely suited for solving unconstrained binary quadratic problems (UBQPs), or QUBO – a simpler version of MIQPs, where $x \in \mathbf{B}^n$. Introducing constraints into QUBO problems is relatively straight-forward, yielding QUBO with penalty quadratic term:

$$x^T Q x + \lambda \sum \big(\max\big(0, b - Ax\big)\big)^2, \tag{2}$$

where all constraints are in the form $Ax - b \geq 0$. Here, $A \in \mathbb{R}^{m \times n}$ is the constraint-coefficient matrix, $b \in \mathbb{R}^m$ is the vector of right-hand-side constants, and $\lambda > 0$ is the penalty weight (typically chosen large; it may be a scalar or set per constraint). The summation is over the m constraint components.
Integer variables *can* be introduced into the problem using e.g. slack variables – a procedure of rewriting each integer variable in the binary basis. This operation complicates the problem statement substantially, while also increasing the size of the problem matrix. Here we work exclusively with QUBO formulation with quadratic penalty terms.
In this contribution we explore solutions proposed in the provided GitHub repository, utilizing state-of-the-art quantum computing resources for solving an approximation of an index-tracking quadratic problem.

1.1 The Optimization Model

For the purpose of this exploration we employ two approaches:

1. Explore solutions to the original simplified 31 bond `.lp` model (provided GitHub repository)
2. Use synthetic model generation from `model_generator3` (created by us)

The original problem statement is known, while the synthetic problem generators work according to the following problem statement:

1.1.1 Input Parameters

- **A set of securities** C with:
  - p_c: market price of bond c
  - m_c: minimum trade size of bond c
  - M_c: maximum trade size of bond c
  - i_c: current basket inventory of bond c
  - δ_c: minimum increment for trading bond c
- **A set of risk buckets** L:
  - K_ℓ is the (possibly overlapping) subset of bonds in bucket ℓ
- **A set of characteristics** J with, for each characteristic j and bucket ℓ:
  - $K_{\ell,j}^{\text{target}}$: target value of characteristic j in bucket ℓ
  - $b_{\ell,j}^{\text{up}}$, $b_{\ell,j}^{\text{low}}$: upper and lower guardrails on the *value-based* aggregate of characteristic j in bucket ℓ
  - $\beta_{c,j}$: contribution of one *unit* of bond c to characteristic j
  - $K_{\ell,j}^{\text{up}}$, $K_{\ell,j}^{\text{low}}$: upper and lower guardrails on the *count-based* (binary) aggregate of characteristic j in bucket ℓ

- **Global parameters**
  - N: maximum number of distinct bonds allowed in the portfolio
  - $\min RC,\ \max RC$: minimum and maximum residual cash-flow targets (scaled by the basket market value MV^b)

1.1.2 Decision Variables

- $y_c \in \{0, 1\}$: indicates whether bond c is included in the portfolio

Note: We do *not* treat the trade size x_c as a free variable. Instead we fix it to the midpoint of the feasible interval whenever $y_c = 1$:

$$x_c \;=\; \frac{m_c \;+\; \min\{M_c,\, i_c\}}{2\,\delta_c}\; y_c.$$

1.1.3 Constraints

1. Maximum number of bonds in the portfolio

$$\sum_{c \in C} y_c \;\le\; N.$$

2. Residual cash-flow constraint

$$\frac{\min RC}{MV^b} \;\le\; \sum_{c \in C} \frac{p_c}{100\,MV^b}\,\delta_c\,x_c \;\le\; \frac{\max RC}{MV^b}.$$

3. Value-based guardrails on each characteristic j in each bucket ℓ
   - Upper bound: $\sum_{c \in K_\ell} \frac{p_c}{100\,MV^b}\,\delta_c\,\beta_{c,j}\,x_c \;\le\; b^{\text{up}}_{\ell,j} \quad \forall\, j \in J,\ \ell \in L.$
   - Lower bound: $\sum_{c \in K_\ell} \frac{p_c}{100\,MV^b}\,\delta_c\,\beta_{c,j}\,x_c \;\ge\; b^{\text{low}}_{\ell,j} \quad \forall\, j \in J,\ \ell \in L.$
4. Count-based (binary) guardrails on each characteristic j in each bucket ℓ
   - Upper bound: $\sum_{c \in K_\ell} \beta_{c,j}\,y_c \;\le\; K^{\text{up}}_{\ell,j} \quad \forall\, j \in J,\ \ell \in L.$
   - Lower bound: $\sum_{c \in K_\ell} \beta_{c,j}\,y_c \;\ge\; K^{\text{low}}_{\ell,j} \quad \forall\, j \in J,\ \ell \in L.$

1.1.4 Objective Function

Match each characteristic j in each bucket ℓ to its target by minimizing the weighted squared deviations:

$$\min_{y} \quad \sum_{\ell \in L} \sum_{j \in J} \rho_j \left(\sum_{c \in K_\ell} \beta_{c,j}\,x_c \;-\; K^{\text{target}}_{\ell,j} \right)^2,$$

where ρ_j is the weight assigned to characteristic j.

1.2 Synthetic Model Statement

The model employed here to benchmark the simulated quantum solution makes use of a reduced number of parameters. The model generator uses a hash function to generate a random feasible QUBO for a selected number of bonds, drawing values for the following characteristics and guardrails from realistic ranges.

Bond Characteristics

1. Duration (1–15 years)

What it is: Measures a bond's price sensitivity to interest rate changes. It is approximately the percentage change in bond price for a 1% change in yield.
Why relevant: Duration is the fundamental risk measure in fixed income. Portfolios need to be controlled for interest rate risk by targeting specific duration exposures.
Range justification:

- Short-term bonds: 1–3 years (treasury bills, short notes)
- Medium-term: 3–10 years (corporate bonds)
- Long-term: 10–15 years (long government bonds, some corporates)
- 15 years is a reasonable upper bound since very long bonds (20–30 years) are less common in typical portfolios

2. Credit Risk (0–1 scale)

What it is: A normalized score representing default probability or credit quality (0 = highest quality like AAA, 1 = highest risk like junk bonds).
Why relevant: Credit risk is the second most important factor in bond investing. Models need to balance yield pickup against default risk.
Range justification:

- 0.0–0.2: Investment grade (AAA to BBB)
- 0.2–0.6: Lower investment grade to high yield
- 0.6–1.0: Speculative/junk bonds
- This gives a reasonable distribution across the credit spectrum

3. Liquidity (0.1–1.0 scale)

What it is: How easily a bond can be traded without significant price impact (1.0 = highly liquid like Treasury bonds, 0.1 = illiquid corporate bonds).
Why relevant: Liquidity affects transaction costs and the ability to rebalance. Less liquid bonds are thought to offer higher yields but increase portfolio risk.
Range justification:

- 0.8–1.0: Government bonds, large corporate issues
- 0.5–0.8: Medium-sized corporate bonds
- 0.1–0.5: Small issues, emerging market bonds
- Minimum of 0.1 ensures all bonds have some tradability

Bond Parameters

1. Price (95–105)

Reasoning: Bond prices are assumed trade around par (100) with reasonable deviations:

- Below 100: Trading at discount (higher yield)
- Above 100: Trading at premium (lower yield)
- ± 5 points represent assumed market variations without extreme distress or premium scenarios

2. Trade Sizes

- **Min trade**: 10,000–50,000 (assumed institutional minimums)
- **Max trade**: 2–10× minimum (reflects available inventory)
- **Inventory**: 0.5–1.5× max trade (assumed dealer inventory levels)
- **Increment**: 1,000–5,000

3. Cash Flow Constraints

- **Min RC**: 2–5% of basket value (reasonable minimum cash generation)
- **Max RC**: 8–12% of basket value (prevents over-concentration in high-yielding bonds)

Portfolio-Level Parameters

1. Basket Market Value ($1M)

- Large enough to be realistic for institutional portfolios
- Round number that makes percentage calculations intuitive
- Allows for meaningful bond positions

2. Maximum Bonds (30–80% of universe)

- Prevents over-diversification (holding tiny positions in everything)
- Forces meaningful selection decisions
- Reflects real-world portfolio management constraints (due diligence, monitoring costs)

3. Characteristic Weights (0.1–2.0)

- Ensures all characteristics have meaningful impact
- Allows for different priority levels
- Range prevents any single characteristic from completely dominating

4. Target Ranges

- **Duration targets**: 3–12 years (covers short to long-term strategies)
- **Credit targets**: 0.2–0.8 (avoids extreme AAA-only or junk-only portfolios)
- **Liquidity targets**: 0.3–0.9 (realistic liquidity management)

1.3 Model Summary

In conclusion, the model generator automates production of synthetic index-tracking QUBOs based on the formulation provided in the problem statement. For numerical stability the prices are given in lots and characteristics are normalized. The input data for a model includes:

1. List of bonds with: `BondId, Price, MinTrade, MaxTrade, Inventory, Increment, Bucket (GOVT, CORP), BetaDurationPerLot, BetaCreditRiskPerLot, BetaLiquidityPerLot`
2. Normalized targets for both `GOVT, CORP`: `Duration, CreditRisk, Liquidity`
3. Dimensionless weights for `Duration, CreditRisk, Liquidity`
4. Parameters: `MVBasketLots, MinRCLots, MaxRCLots, MaxBonds`
5. Normalized Guardrails or both `GOVT, CORP`: `Duration, CreditRisk, Liquidity`

The model generator creates a pseudo-random feasible problem of a type 'include a bond in the basket or not' for N bonds, subject to 15 constraints in total: maximum number of bonds included, minimum and maximum cash flow; lower and upper guardrails for duration, credit risk, liquidity of both GOVT and CORP bonds; while minimizing the deviation from targets for characteristics values. The model generator is thought out as a robust piece of code capable of generating realistic, feasible problems with non-trivial constraints and multiple feasible sub-optimal solutions. The rich problem structure guaranteed by our code lends itself elegantly to detailed quantum solution benchmarking, as explored in the following sections.

1.4 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is a hybrid quantum–classical algorithm designed to approximate the ground-state energy of a given Hamiltonian $\hat{H}$, which can represent either a quantum system in physics or, after suitable reformulation, a discrete optimization problem such as QUBO.
It is based on the variational principle, which guarantees that for any normalized trial state $|\psi(\theta)\rangle$ parameterized by $\theta \in \mathbb{R}^p$:

$$E(\theta) = \langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle \geq E_0 \tag{3}$$

where E_0 is the true ground-state energy of $\hat{H}$.
In VQE, the workflow proceeds as follows:

1. The problem Hamiltonian is expressed as a weighted sum of tensor products of Pauli operators. For combinatorial optimization, the QUBO cost function $x^T Q x$ can be directly mapped to $\hat{H}$ via binary-to-qubit encoding, e.g. $x_i \mapsto \frac{1}{2}(1 - Z_i)$, making VQE a practical approach for solving constrained optimization problems on noisy intermediate-scale quantum (NISQ) devices.
2. Ansatz is prepared using a parameterized quantum circuit $U(\theta)$, i.e., $|\psi(\theta)\rangle = U(\theta)\,|0\rangle^{\otimes n}$.
3. The quantum computer estimates $E(\theta)$ by measuring the expectation value of the Hamiltonian.
4. Classical optimizer updates θ to minimize the energy:

$$\theta^* = \text{argmin}_\theta\, E(\theta). \tag{4}$$

This loop continues until convergence criteria (energy tolerance or iteration limit) are met.

2 Exploring the codebase – 31 bonds

Based on the evaluations (provided in the files `analysis31_one_run.ipynb`, `analysis31_one_run.ipynb`, `analysis31_step3_TwoLocal.ipynb`, `analysis31_step3_bfcd.ipynb`), we reviewed the performance of two ansatz architectures within the VQE framework – TwoLocal and bfcd – for the 31-bond optimization problem. These evaluations include variations in:

- Repetition depth $r \in \{1, 2, 3\}$, which defines the number of rotation block + entanglement block in the ansatz circuit. Increasing r increases the expressibility of the ansatz at the cost of more circuit depth,
- The CVaR aggregation parameter $\alpha \in \{0.1, 0.15, 0.2\}$, which defines the fraction of the lowest-cost measurement outcomes averaged to compute the objective value.

Key insights from these results:

- The bfcd ansatz consistently achieved faster convergence compared to TwoLocal. This is likely due to its problem-specific design, with an entanglement structure more closely aligned with the cost landscape.
- Across both ansatzes, the effect of varying r and α appeared to be relatively minor within the tested ranges, based on the existing output. This suggests a degree of robustness in performance.
- The TwoLocal ansatz includes an additional hyperparameter: the entanglement topology (full vs. bilinear). According to the results, bilinear entanglement consistently outperformed the full pattern.

Table 1: Circuit characteristics of the TwoLocal and bfcd ansatzes for the 31-bond problem with repetition depth $r = 2$, bilinear entanglement, and CVaR parameter $\alpha = 0.1$. Depth is the total circuit depth after transpilation. Two-qubit depth is the circuit depth restricted to layers containing only two-qubit gates. Parameters is the number of variational parameters. Gate counts list the number of each gate type in the transpiled ansatz.

	Depth	Two-qubits Depth	Qubits	Parameters	Gate Counts
TwoLocal	8	4	31	93	'ry': 93 'cz': 60
bfcd	19	8	31	182	'rx': 180 'rzz': 120 'ry': 62

Based on this analysis, we focused our further exploration on the TwoLocal and bfcd ansatzes, using a fixed repetition depth $r = 2$, a fixed optimizer parameter $\alpha = 0.1$, and bilinear entanglement for both ansatzes. The characteristics of each ansatz—circuit depth, number of two-qubit gates, total number of qubits, gate counts, and number of parameters—are listed in Table 1.
We then evaluated how the penalty value and the choice of parameter initialization (fixed $\pi/3$ vs. random) influence the performance of the optimization. The benchmarking results for the original 31-bond problem are summarized in Tables 2 and 3.

2.1 Penalty term

Files: `penalty_analysis.ipynb`, `sbo_steps1to3_penalty.py`

Given this fixed configuration, we next turned our attention to the penalty term used to embed linear constraints into the objective function. Since the optimization process converts a constrained binary quadratic program into an unconstrained one, the penalty parameter λ determines the weight of constraint violation in the overall cost function. Choosing a value that is too low can result in constraint violations, while a value that is too high may lead to poor optimization dynamics or slower convergence.
We evaluated the impact of different penalty values on optimization performance. The tested penalty values were: $\lambda = 1.1, 1.5, 2.0, 2.5$.
The average trajectory of the objective function during the optimization process is presented in Fig. 1 for both the TwoLocal and bfcd ansatzes. For each penalty value, the optimization was run 10 times. The resulting cost function values were averaged across iterations, and the standard deviation is shown as a shaded band, indicating the variability of the optimization process. Table 2 summarizes the performance metrics of the optimization for each penalty value. For both ansatzes, the lowest penalty value ($\lambda = 1.1$) frequently led to better convergence and higher similarity to the reference solution, while maintaining constraint satisfaction. Higher penalties led to longer runtimes and less stable convergence behavior.

2.2 Circuit initialization

Files: `random_initialization_analysis.ipynb`, `sbo_steps1to3_randominitialization.py`

To examine the influence of parameter initialization, we compared the commonly used uniform initialization where all parameters are set to $\theta = \pi/3$ (used in all previous simulations) against random initialization schemes. The goal was to test whether random initialization of ansatz parameters improves the convergence and solution quality in the VQE procedure.
This experiment was conducted with fixed settings: repetition depth $r = 2$, CVaR parameter $\alpha = 0.1$, bilinear entanglement pattern, and penalty value $\lambda = 1.1$. The convergence trajectories in Fig. 2 represent averages over 10 optimization runs using the same randomly initialized parameter set. This process was repeated multiple times with different random initializations. Table 3 reports the performance metrics corresponding to each initialization batch.
For both ansatzes, random initialization sometimes yielded comparable or slightly better solutions in individual runs. However, it also introduced greater variance in performance, particularly for the TwoLocal

Table 2: Performance metrics for TwoLocal ansatz and bfcd ansatz under different penalty values λ. Symbol used: t_{avg} – average runtime of the optimization for a single run, hits – number of runs (out of 10) where the minimum objective value exactly matched the optimal value computed using CPLEX, $f_{\min}$ – the best (lowest) objective value found across all 10 runs, c_{sat} whether the solutions corresponding to $f_{\min}$ satisfied all constraints, $\bar{f}_{\min}$ – the average of the minimum objective values over all runs, σ_f – standard deviation of the minimum objective values, $\bar{d}_H$ – the average Hamming similarity between the best-found binary solution and the reference solution over 10 runs. This metric quantifies how close the binary decision vector is to the optimal one, with 1.0 indicating an exact match.

	TwoLocal				bfcd			
λ	1.1	1.5	2.0	2.5	1.1	1.5	2.0	2.5
Runs	10	10	10	10	10	10	10	10
t_{avr} [s]	7786	6604	7031	6402	15330	20364	23443	22279
Hits	1	1	0	1	3	2	0	0
$f_{\min}^{\text{CPLEX}}$	40.2939							
$f_{\min}$	40.2939	40.2939	40.3103	40.2939	40.2939	40.2939	40.3028	40.3198
c_{sat}	80%	50%	50%	80%	70%	90%	80%	60%
$\bar{f}_{\min}$	40.8646	41.6777	40.9113	41.1159	40.7471	40.7880	41.0659	40.8473
σ_f	0.4224	0.9197	0.4189	0.8343	0.2859	0.3780	0.4989	0.3381
$\bar{d}_H$	0.8	0.7193	0.7612	0.7483	0.8645	0.8387	0.7161	0.6387

ansatz. The $\pi/3$ initialization showed more consistent behavior across runs. For bfcd, random initialization remained relatively robust and even marginally outperformed $\pi/3$ in some cases.

2.3 Solution feasibility

File: `feasibility.ipynb`

One of the defining properties of the optimization problem is the presence of constraints. It is of high value to inspect if a proposed solution violates any of the constraints imposed by the problem statement. We built a module `model_validator` on top of `docplex` to investigate solution feasibility. We resorted to binary classification c_i=0/1 assigning 0 if any of the constraints are violated by the proposed solution. This classification is represented by c_{sat}-score for any given series of N runs:

$$c_{\text{sat}} = \frac{\sum_{i=1}^{N} c_i}{N}, \tag{5}$$

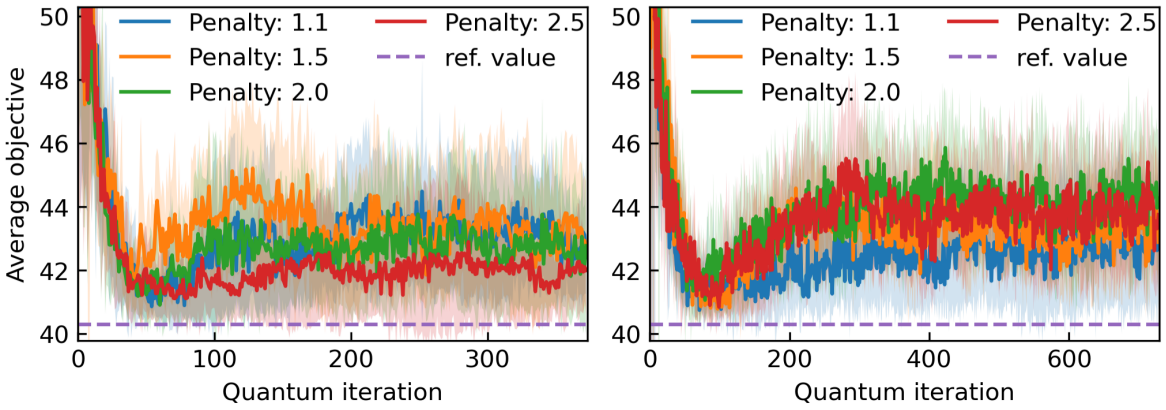

Figure 1: Average trajectory of the objective function during the optimization process for different values of the penalty for TwoLocal andatz (left) and bfcd ansatz (right). The ansatz parameters: $r = 2$, $\alpha = 0.1$, and bilinear entaglement.

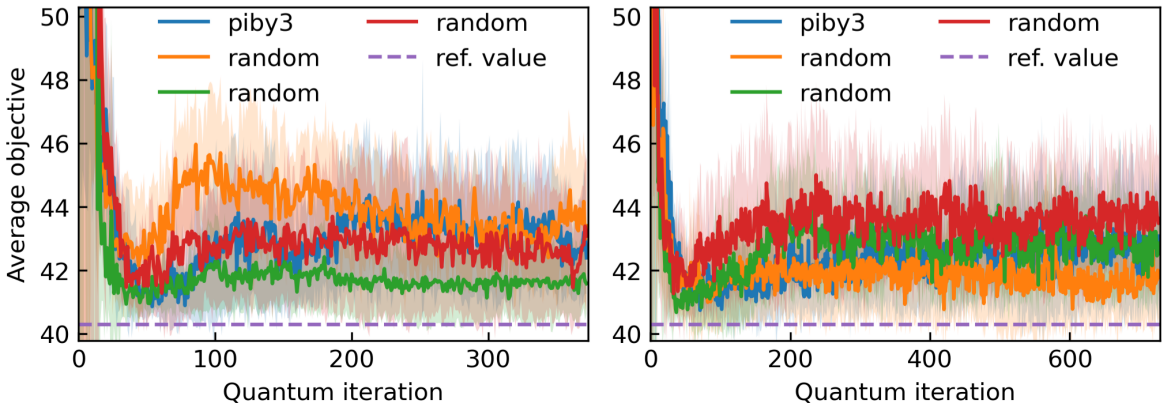

Figure 2: Average objective function value during optimization with different initialization schemes for TwoLocal ansatz (left) and bfcd ansatz (right). Each curve represents the average of 10 runs. Fixed parameters: $r = 2$, $\alpha = 0.1$, $\lambda = 1.1$, bilinear entanglement.

Table 3: Performance metrics for different initialization schemes. The notation is the same as in Table 2.

	TwoLocal				bfcd			
Initialization	$\pi/3$	random	random	random	$\pi/3$	random	random	random
Runs	10	10	10	10	10	10	10	10
t_{avr} [s]	7786	5536	5479	5931	15330	10879	11885	12886
Hits	1	1	1	1	3	3	1	0
$f_{\text{min}}^{\text{CPLEX}}$	40.2939							
f_{min}	40.2939	40.2939	40.2939	40.2939	40.2939	40.2939	40.2939	40.3014
c_{sat}	80%	70%	30%	50%	70%	40%	60%	40%
f_{min}	40.8646	42.1576	40.9444	41.2846	40.7471	40.7740	40.6824	41.2730
σ_f	0.4224	2.1133	0.7049	0.7252	0.2859	0.2372	0.2564	1.2133
d_H	0.8	0.7870	0.6935	0.7483	0.8645	0.8516	0.7612	0.7612

with the most desirable outcome being $c_{\text{sat}} = 100\%$.
Crucially, our benchmarking on the original 31 bond problem reveals that there is a non-negligible portion of runs which end with an infeasible solution. Interestingly enough, Tables 2, 3 show that feasibility of the solution is the value affected most by tinkering with solver parameters. Furthermore, c_{sat}=100% for all but one synthetic data runs. This perhaps suggests that our synthetic models cannot fully capture the numerical complexity of the original problem. The results from this section show how important it is to thoroughly benchmark a quantum solver, underpinning the need for a feasibility checker. Our rudimentary score should in the future be replaced by a more comprehensive metric that tests not only if, but also by how much given constraints have been breached.

3 Different Number of Bonds

Now we turn our attention to the studies performed using synthetic data (see Section 1.3). We utilize `model_generator` to probe how the hybrid solver solutions scale with the problem size by building a 10-attempt-statistics for random 15-, 20-, 25- and 30-bond problems. Moreover, we leverage our ability to mass-produce problems to build more robust aggregate statistics by comparing the hybrid solution's performance on 100 attempts to solve the same 30-bond random problem with its performance on 100 different random 30-bond problems.

3.1 Scaling

Files: `bonds_analysis_scaling.ipynb`, `sbo_steps1to3_scaling_distribution.py`

The results from solving 15-, 20-, 25-, and 30-bond problems with TwoLocal and bfcd with $\alpha = 0.1$,

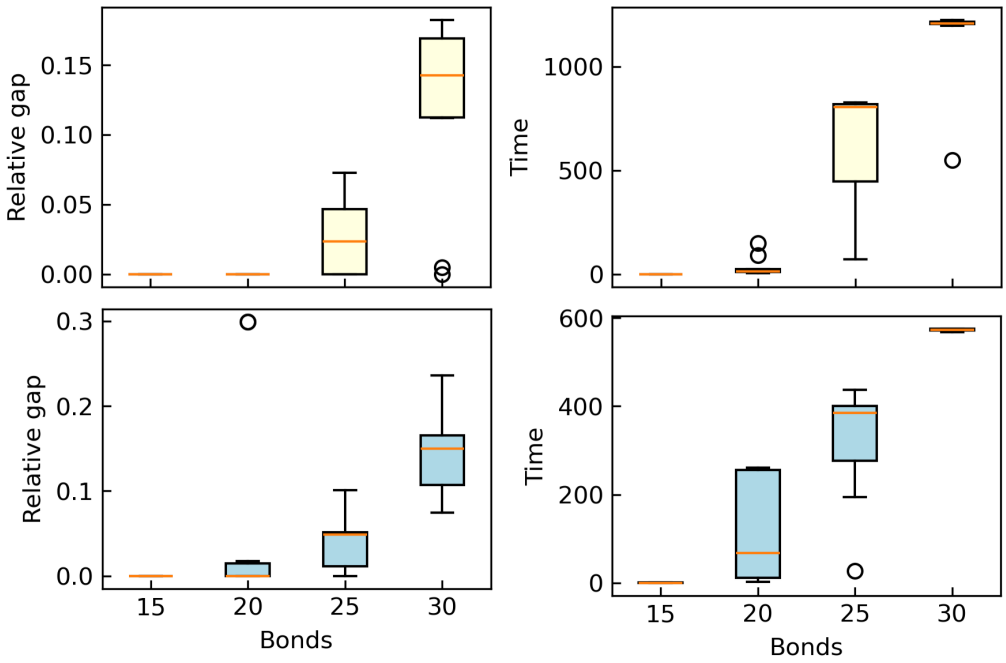

Figure 3: Scaling analysis of the hybrid solver for portfolio optimization. Top: TwoLocal ansatz with $r = 2$, $\alpha = 0.1$, bilinear entanglement. Bottom: bfcd ansatz with $r = 2$, $\alpha = 0.1$, bilinear entanglement. Each bar corresponds to one of 10 independent runs for problem sizes of 15, 20, 25, and 30 bonds, using $\pi/3$ initialization. Reported metrics are the relative optimality gap (left panel) and the total solution time (in seconds; right panel).

Table 4: Circuit characteristics of the TwoLocal and bfcd ansatzes for problem instances generated with varying numbers of bonds. The notation is the same as in Table 1.

	TwoLocal				bfcd			
Bonds	15	20	25	30	15	20	25	30
Depth	8	8	8	8	19	19	19	19
Two-qubit Depth	4	4	4	4	8	8	8	8
Qubits	15	20	25	30	15	20	25	30
Parameters	45	60	75	90	86	116	146	176
Gate Counts	'ry', 45 'cz', 28	'ry', 60 'cz', 38	'ry', 75 'cz', 48	'ry', 90 'cz', 58	'rx', 84 'rzz', 56 'ry', 30	'rx', 116 'rzz', 76 'ry', 40	'rx', 144 'rzz', 96 'ry', 50	'rx', 176 'rzz', 116 'ry', 60

$r = 2$, bilinear entanglement, and $\pi/3$ initialization are summarized in Fig. 3 and Tab. 4. The bar plots show the results of 10 attempts to solve the same problem. For both Ansatze, solution time and relative gap increase with problem size. While the increase in time is expected, the more concerning trend is the growing deviation from the optimal objective. For small-sized binary problems, even straightforward enumeration is a viable solution strategy, so the hybrid solver's ability to reach the optimal objective is reassuring. However, since the simulator's qubit limit is 31, we could not test the hybrid solver's performance on larger problems, and the onset of this trend is nonetheless worrisome.

3.2 Statistics

Files: `bonds_analysis_distribution.ipynb`, `sbo_steps1to3_scaling_distribution.py`

Next, we investigate the results of an insightful study, i.e. running a more detailed statistics on 30 bonds: 100 attempts at solving the same problem versus a single attempt at solving 100 different problems once. All runs for the following solver configuration: TwoLocal, $\alpha = 0.1$, $r = 2$, bilinear entanglement, and $\pi/3$ initialization. The results are show in Fig. 4.

The first landmark result from this dataset was already mentioned: only a single of these 200 runs produced an infeasible solution, marking the fact that indeed `problem_generator` does not generate problems equivalent to the original 31-bond one.
Comparing relative gap results in Fig. 4 (a) and Fig. 4 (c), we note several points of interest:

- (c) — The majority (∼50%) of possible 31-bond problems have single-shot optimal solutions; however, for about 5% of problems, the proposed objective is more than 50% off from the optimum, the problem landscape has a very long tail.
- (a) — The problem selected for 100 attempts is relatively uncommon, with an objective value on average ∼20% worse than the optimum.
- (a) — There is a non-negligible variance in the relative gap distribution, and "hard" problems appear to be rarely ($< 5\%$ of all attempts) solved optimally.

Execution-time-wise Fig. 4 (b,d) we note a pronounced mean time with very long tails. These results however are of little interest as they are dominated by simulation overhead. We include them for completeness as these should be present when benchmarking an actual quantum hardware runs.
All of the comparisons made above are made easily readable in boxplots (e,f).

In summary, the results presented here warrant cautious pessimism, as there appears to be a long tail of problems with identical structure but differing numerical values that remain difficult for the hybrid solver to tackle.

4 Heat Exchange Ansatz

Reference:
`Soyoung Shin, Ha Eum Kim, Hyeonjun Yeo, Kabgyun Jeong, Wonho Jhe, Jaewan Kim`
`Designing Minimalistic Variational Quantum Ansatz Inspired by Algorithmic Cooling,`
`arXiv:2501.16776 (Jan 2025)`
File: `heat-exchange-ansatz.ipynb`

To explore new directions in ansatz design, we investigated the Heat Exchange (HE) ansatz, recently proposed in [Shin2025]. This minimalistic variational ansatz is inspired by algorithmic cooling techniques—specifically, *heat-bath algorithmic cooling* (HBAC)—and was originally designed to improve performance on combinatorial optimization problems such as MAXCUT.
Unlike HBAC, which requires repeated resets of auxiliary (bath) qubits via coupling to an external thermal reservoir, the HE ansatz is fully coherent and unitary. It enables entropy redistribution between problem qubits and bath qubits without the need for resets, making it highly suitable for implementation on near-term (NISQ) quantum devices.
Key Characteristics:

- HE uses parameterized XX+YY gates to emulate thermalization-like behavior, swapping population between problem and bath qubits.
- Bath qubits are initialized (typically to $|1\rangle$) once and used coherently throughout.
- Each problem qubit only needs to interact with its corresponding bath qubit. This makes the ansatz hardware-friendly and efficient for devices with limited connectivity (e.g., IBM's quantum processors).
- The ansatz has been shown to outperform QAOA and hardware-efficient ansatzes on weighted MAXCUT problems, which are structurally similar to our portfolio optimization setting.

Due to time constraints within the project, we did not integrate the HE ansatz into the full VQE pipeline. However, we implemented the circuit independently in Qiskit, verified its structure, and plan to integrate it into our optimization routines as part of future work. We believe this ansatz has high potential for problems in quantum optimization due to its: simplicity and low depth, compatibility with QUBO-style cost functions, efficient hardware mapping.

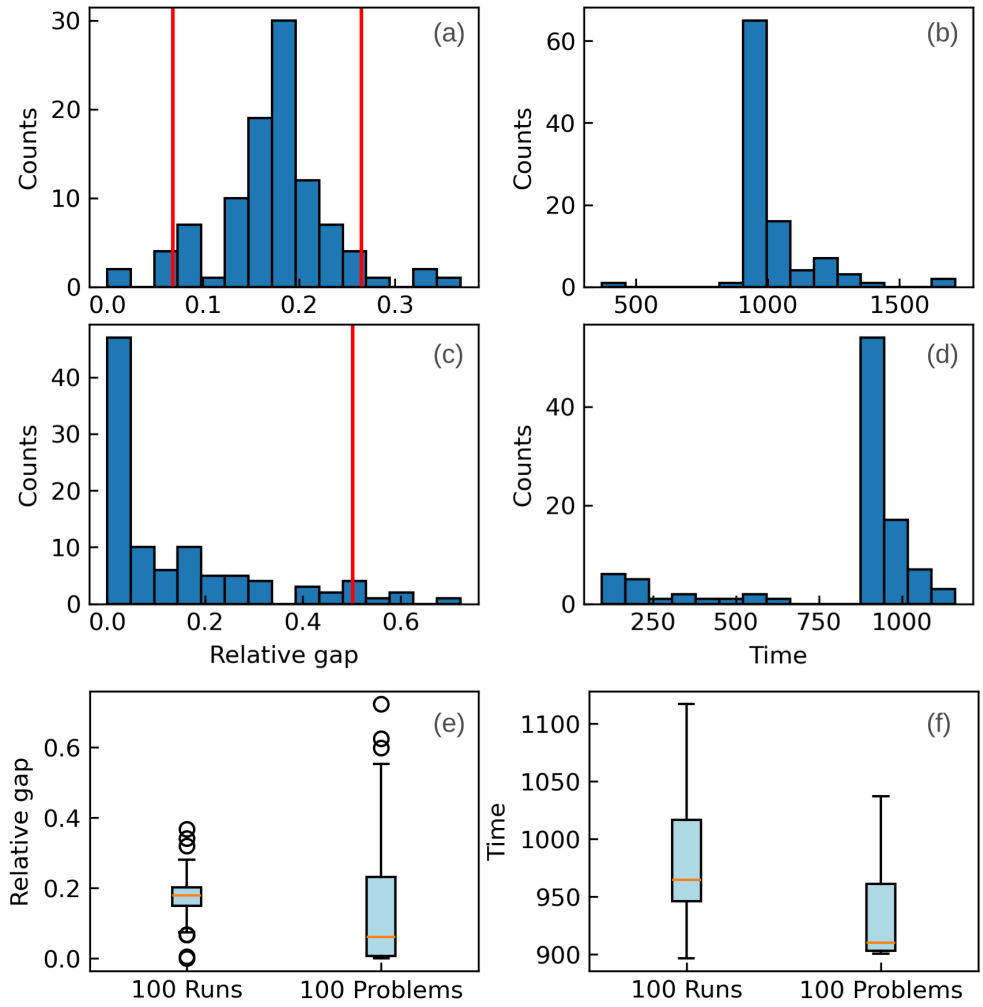

Figure 4: Performance distribution analysis for the TwoLocal ansatz with $r = 2$, $\alpha = 0.1$, bilinear entanglement, and $\pi/3$ initialization. Top (a,b): histograms from 100 runs of the *same* problem instance — (a) relative optimality gap, with red lines indicating the 5th and 95th percentiles; (b) execution time. Middle (c,d): histograms from 100 *different* problem instances (one run each) — (c) relative optimality gap, with the red line indicating the 95th percentile; (d) execution time. Bottom (e,f): boxplot comparisons of relative gap and execution time across the two settings above.

5 Summary and Future directions

Our work aims to benchmark a state-of-the-art hybrid QUBO solver. Since we relied on simulations rather than quantum hardware, this study should be regarded as a pilot. Additionally, we need to stress that we did not use any data post-processing and show almost raw results, other than averages in boxplots. The extensive statistics we gathered shows a long tail of difficult-to-solve problems – an issue, that we show in our scaling analysis, bound to only grow more severe with growing portfolio sizes.

Our benchmarking cautiously suggests the need for a different quantum solver approach on QISE — beginning with a search for a more suitable Ansatz and potentially extending to a ground-up redesign of the solver architecture.

Future goal: integrate the HE ansatz into the VQE framework already developed for portfolio optimization and benchmark its performance against TwoLocal and bfcd Ansatze.

Benchmarking:

1. Implement a comprehensive constraint-violation metric to differentiate between benign and severe

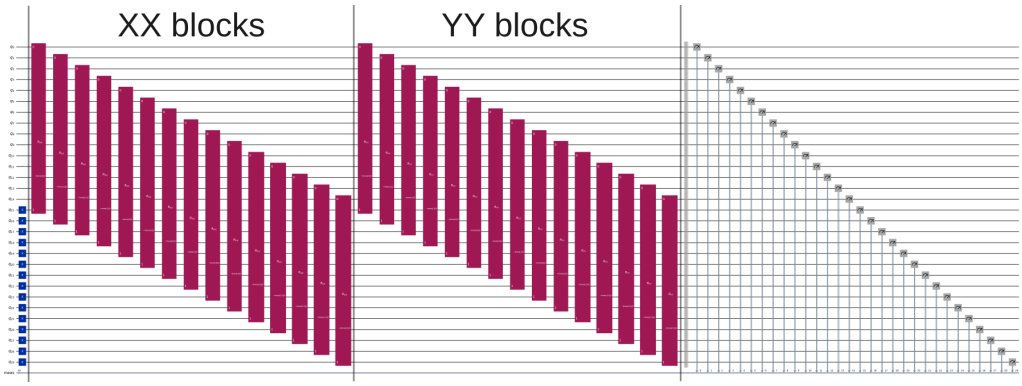

Figure 5: Heat Exchange ansatz implementation in Qiskit

breaches – "slightly wrong" fast solutions could be considered desirable over optimal solutions which do not show quantum advantage

2. Run scaling benchmarks and compare execution times on quantum hardware – test scaling beyond qubit simulation limits and actual execution times

References

[Shin2025] S. Shin, et. all
Designing Minimalistic Variational Quantum Ansatz Inspired by Algorithmic Cooling, arXiv:2501.16776 (Jan 2025)