Journal of
Quantum Computing

Tech Science Press

Doi:10.32604/jqc.2025.068127

ARTICLE

Investigating Techniques to Optimise the Layout of Turbines in a Windfarm Using a Quantum Computer

James Hancock*, **Matthew Craven**, **Craig McNeile and Davide Vadacchino**

Centre for Mathematical Sciences, University of Plymouth, Plymouth, PL4 8AA, UK

*Corresponding Author: James Hancock. Email: james.hancock@plymouth.ac.uk

Received: 21 May 2025; Accepted: 21 July 2025; Published: 11 August 2025

ABSTRACT: This paper investigates Windfarm Layout Optimization (WFLO), where we formulate turbine placement considering wake effects as a Quadratic Unconstrained Binary Optimization (QUBO) problem. Wind energy plays a critical role in the transition toward sustainable power systems, but the optimal placement of turbines remains a challenging combinatorial problem due to complex wake interactions. With recent advances in quantum computing, there is growing interest in exploring whether hybrid quantum-classical methods can provide advantages for such computationally intensive tasks. We investigate solving the resulting QUBO problem using the Variational Quantum Eigensolver (VQE) implemented on Qiskit's quantum computer simulator, employing a quantum noise-free, gate-based circuit model. Three classical optimizers are discussed, with a detailed analysis of the two most effective approaches: Constrained Optimization BY Linear Approximation (COBYLA) and Bayesian Optimization (BO). We compare these simulated quantum results with two established classical optimization methods: Simulated Annealing (SA) and the Gurobi solver. The study focuses on 4 × 4 grid configurations (requiring 16 qubits), providing insights into near-term quantum algorithm applicability for renewable energy optimization.

KEYWORDS: Quantum computing; QUBO; windfarm layout optimization; VQE

1 Introduction

The climate crisis and ongoing efforts to reduce emissions provide strong motivation for maximizing energy extraction from renewable sources. Utilizing these energy sources minimizes environmental impact and promotes a sustainable future.

One key renewable energy source is wind. There are many challenges to making it an economically viable and reliable energy source [1]. In this paper, we focus on using quantum computers to solve one specific problem towards maximizing the energy extracted from the wind. There are currently limitations on the available quantum computing hardware, which we will discuss later. Quantum computing applications in renewable energy have been reviewed in Ref. [2,3]. Recent work has specifically addressed quantum optimization methods [4].

Energy from the wind is extracted using wind turbines. Typically, wind turbines are combined to form windfarms. The spatial layout of the turbines in the windfarm will change the maximum amount of energy that the windfarm can produce. The use of optimization techniques for wind farm layout optimization dates back decades [5], with modern approaches continuing to develop [6].

The exponential growth in optimization complexity with increasing turbine count motivates the development of improved algorithms for optimal turbine placement. Recently, one form of the Windfarm Layout Optimization (WFLO) problem has been rewritten as a Quadratic Unconstrained Binary Optimization (QUBO) problem [7] that can be solved on a quantum computer.

There are currently two dominant types of quantum computers. One is based on quantum circuits. For example, IBM and Google have constructed quantum computers based on this paradigm. One way to solve QUBO problems using a quantum circuit computer is to use a Variational Quantum Eigensolver (VQE).

Other types of quantum computers are Quantum Annealers (QAs) [8]. QAs are built to solve QUBO problems. D-WAVE's Advantage is the largest real-world QA, having 5000+ qubits (although not all connected) [9]. A new system currently under development, promising over 7000 qubits [10], is expected to be available in the coming years. Fujitsu's Digital Annealer can also solve QUBO problems using specialized hardware [11].

In this paper, we investigate using quantum optimization algorithms to solve the WFLO problem mapped to a QUBO problem using the `qiskit` package [12] from IBM, running on a classical simulator. We employ the Variational Quantum Eigensolver (VQE), an approach specifically designed for the Noisy Intermediate-Scale Quantum (NISQ) era [13]. This work is a necessary first step to solving the WFLO problem using quantum computers.

Apart from the importance of solving the WFLO problem to produce more electricity from a windfarm, there are other motivations for this study. QUBO problems are a subset of combinatorial optimization problems. Many optimisation problems can be mapped to QUBO. For example, the Maximum Cut (MAXCUT) problem from graph theory can be mapped to a QUBO. The QUBO problems from different applications may be easier or harder to solve using different methods, so it is important to find empirical evidence for the performance required for the solution in these different difficulty settings [14]. Our contribution is to investigate solving the QUBO problem mapped from a WFLO using a simulator of a circuit-based quantum computer.

QUBO problems are one of the main problems that can be solved by annealers such as those sold by D-Wave and Fujitsu [15]. The solution of WFLO problems may be an interesting test case for comparing the performance of adiabatic and circuit-based quantum computers. Previous work has shown the difficulty in comparing the performance of classical computers and adiabatic annealers in solving QUBO problems [16–19].

In this work, we have used the circuit model of quantum computing as the basis of the calculation. Namely, `qiskit`'s gate-based quantum circuit model with superconducting qubit architectures, as typically employed in IBM quantum systems. A key challenge in current quantum computing is the absence of quantum error correction. This necessitates algorithms with inherent resilience to decoherence. In practice, NISQ era algorithms typically adopt hybrid quantum-classical architectures to mitigate these limitations. We limit ourselves to a regime where we do not consider the quantum noise effects, such as decoherence. Instead, we tackle the problem in a statistically noisy setting, providing a necessary baseline for future work incorporating quantum noise effects.

The workflow diagram in Fig. 1 shows our overall process, where the quantum and classical pathways are highlighted.

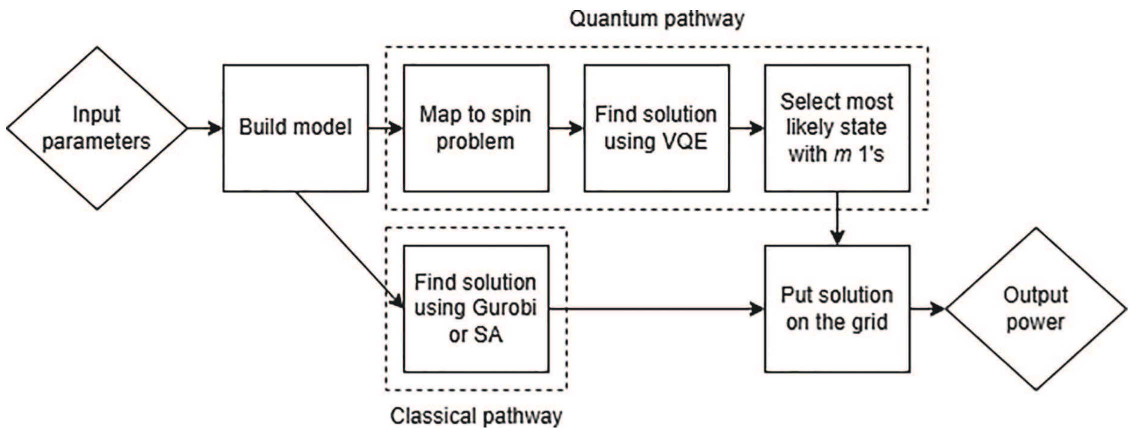

Figure 1: Flowchart of overall process of WFLO

2 Windfarm Layout Optimization

WFLO involves selecting turbine locations to maximize power output under specific constraints. These include: a maximum number of turbines, and enforcing a minimum separation distance between turbines due to rotor interference. We assume turbines can only be placed on discrete grid points.

Turbines positioned downstream (i.e., behind other turbines relative to wind direction) experience reduced power output compared to those exposed to undisturbed wind. This wind speed velocity deficit, known as the turbine wake, creates the fundamental challenge in WFLO: minimizing energy generation losses from wake interactions.

The Sum-of-Squares (SS) model for wake speeds [20] best captures the effect of these wakes. However, the formulation leads to intractable optimization problems. To combat this, we use a simplified model, the Linear Superposition of Wakes (LS) [21]. This allows us to map the problem to a QUBO problem, which can then, in turn, be optimized using various quantum (and classical) methods.

3 Windfarm Layout as a Quadratic Unconstrained Binary Optimization Problem

A QUBO problem is defined as

$$\underset{x}{\operatorname{argmin}}\, f_Q(x), \tag{1}$$

where

$$f_Q(x) := x^T Q x = \sum_{i,j=0}^{q} Q_{ij} x^i x^j \tag{2}$$

for $x \in \mathbb{B}^q = \{0,1\}^q$.

Many combinatorial optimization problems can be formulated as QUBO problems. QUBO is a class of Nondeterministic Polynomial Time (NP)-hard problems [22]. While quantum computers offer ways to find high-quality solutions to QUBO problems, finding guaranteed optimal solutions is not expected to be in the Bounded-Error Quantum Polynomial Time (BQP) complexity class for general cases. QUBO problems, by definition, are formulated in an unconstrained way. However, constraints can be included by introducing penalties in the form of large additive constants when the conditions are violated. We follow the mapping of the WFLO problem to a QUBO problem that was developed by Senderovich et al. in [7].

We model the wind farm terrain as a square grid with side length l_{grid}, allowing for l^2_{grid} potential turbine placement locations. These will be labelled as they are in Fig. 2. The indicator of the presence of a turbine on site q is x^q, where $q \in \{1, \dots l^2_{grid}\}$. Cartesian coordinates of the grid are shown in Fig. 3.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 2: Labelling of sites on a $l_{grid} = 4$ windfarm grid

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

Figure 3: Coordinates of sites on a $l_{grid} = 4$ windfarm grid

A wake can be defined by three parameters: the angle α (degrees) that the wind is coming from with respect to the west counted clockwise, x, the maximum distance that is affected by the wake and r is the radius with which it spreads out per unit distance, away from the turbine causing the wake. These physical factors are problem-specific. For our model, we say that if a grid location is at all within the wake, it is entirely within its wake. We denote a wake starting at position i as $w(i, x, r; \alpha)$.

To visualize the wakes from some turbines in the wind, they are plotted on a $l_{grid} = 10$ grid (larger than that used in the simulations) with the parameters: $x = 3$, $r = 1$. Figs. 4 and 5 show a configuration of the wakes from one turbine. Figs. 6 and 7 show a configuration of the wakes from two turbines. If one of the turbines is in the wake of another turbine, then it will see a reduced wind velocity and thus produce less power.

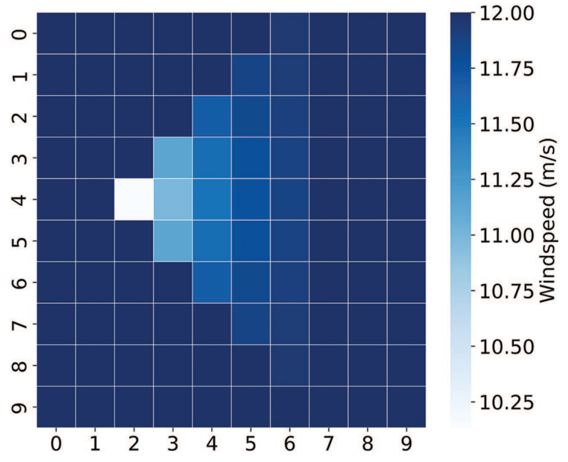

Figure 4: Turbine on location 15 or (2, 4), $D = \{0, 12ms^{-1}, 1\}$

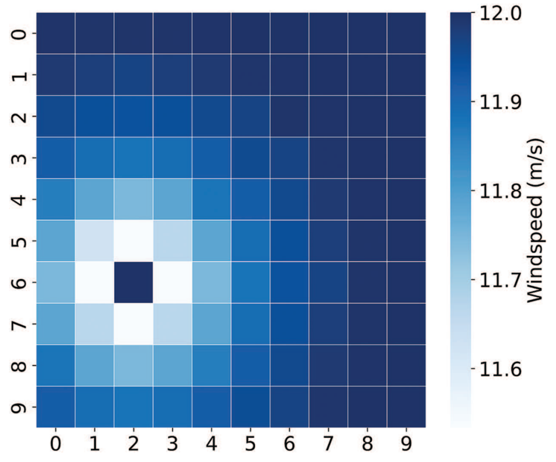

Figure 5: Turbine on location 27 or (2, 6), D = Second Mosetti

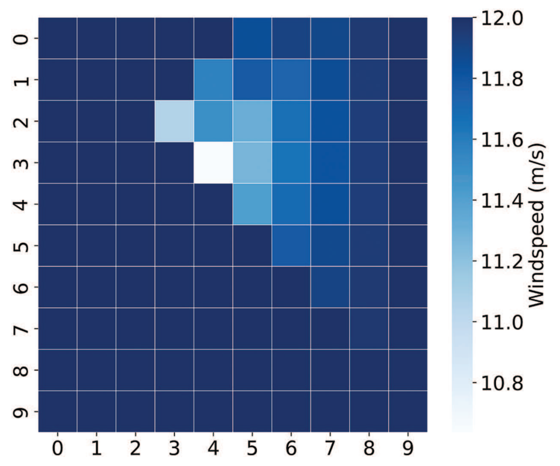

Figure 6: Turbine on location 23 or (2, 2) and 34 or (3, 3), $D = \{0, 12ms^{-1}, 1\}$

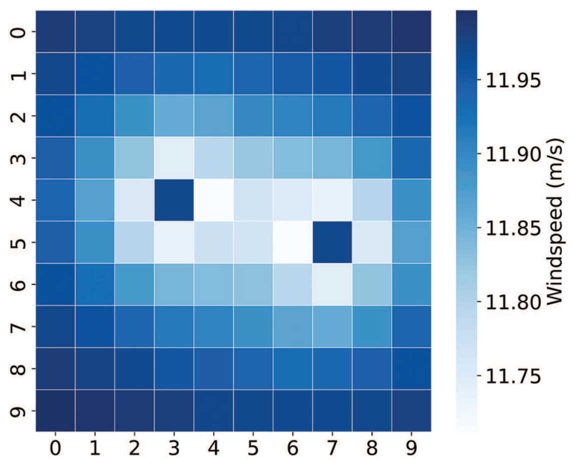

Figure 7: Turbine on location 35 or (3, 4) and 76 or (7, 5), D = Second Mosetti

Model

To ensure that the model is more realistic, the system includes several wind arrangements. Each possible wind arrangement is defined as $d := \{\alpha_d, v_d, p_d\}$. α_d is the angle at which the wind comes in, v_d is the *free wind speed*, i.e., the wind speed that powers a turbine not in the wake of another. p_d is the probability that we would expect to encounter this wind arrangement. We denote the set of possible wind arrangements in our system as D, which we call the *wind regime*. To clarify this, D is a collection of d's, with $\sum_{d \in D} p_d = 1$.

The particular model that we are using is the LS of wakes. This model is useful as it can be transformed into a QUBO problem. The power output for a system of q sites within this model is computed as:

$$E^{\text{LS}} = \sum_{d \in D} \sum_{i=1}^{q} p_d \left[\frac{1}{3} v_d^3 - \sum_{j \in w_i} \frac{1}{3} \left(v_d^3 - u_{ij}^3 \right) \right], \tag{3}$$

where w_i is shorthand for the wake caused by turbine i, and u_{ij} is the reduced wind speed at site j, caused by being in the wake of i. If j is in the wake of i, u_{ij} can be calculated as

$$u_{ij} = v_d \left(1 - \frac{2a}{(1 + \alpha_T (\delta / r)^2)^2} \right). \tag{4}$$

This is the Jansen wake model [23], where a is the axial induction factor, currently set to 0.1, and $\delta = ||i - j||_2$ is the distance between the turbines. By controlling the wake's downstream length (x) and its radial expansion rate (r), we compute α_T using Eq. (5):

$$\alpha_T = \frac{1}{x} \left(r - r_t \right), \tag{5}$$

where r_t denotes the turbine radius. In this work, we set $r_t = 0.33$ (in terms of grid boxes). We note that r_t and a from Eq. (4) carry units; their values are dependent on the physical features of the turbines. The selected values are arbitrary and do not correspond to any physical system, as this represents a simplified theoretical model.

The QUBO formulation for this problem is then:

$$\underset{x}{\operatorname{argmin}}\,(-f(x)), \tag{6}$$

$$f(x) = \sum_{d\in D}\sum_{i=1}^{m} p_d\left[\frac{1}{3}v_d^3 x^i x^i - \sum_{j\in w_i}\frac{1}{3}\left(v_d^3 - u_{ij}^3\right)x^i x^j\right]. \tag{7}$$

This represents the unconstrained problem formulation, where the optimal solution trivially places a turbine at every available site. To enforce practical constraints, we introduce the energetic penalty:

$$g(x;\lambda_1,\lambda_2) = \lambda_1\left(\sum_{i=1}^{l_{grid}^2} x^i - m\right)^2 + \lambda_2 \sum_{||i-j||_2<\xi} x^i x^j \tag{8}$$

The term weighted by λ_1 in Eq. (8) limits the number of total turbines to m; the second term pertains to the minimum distance between the turbines being ξ. The values of λ_1, λ_2 have to be large enough so that the constraints are met. The full QUBO problem is:

$$\underset{x}{\operatorname{argmin}}\,(-f(x) + g(x;\lambda_1,\lambda_2)) \tag{9}$$

The problem from Eq. (9) can then be written in the form of a weight matrix Q from Eq. (2):

$$Q_{ij} = \begin{cases} -\frac{1}{3}\sum_d p_d v_d^3 + \lambda_1(1-2nm) & \text{if } i = j \\ -\frac{1}{3}\sum_d p_d\left(v_d^3 - u_{ij}^3\right) + 2\lambda_1 & \text{if } i \neq j \end{cases} \tag{10}$$

Here, we have neglected the proximity constraints term as we do not include this in our simulations. When the problem is formulated, it is a fully dense Q matrix. We note here that $q = l_{grid}^2$ is the number of variables/qubits.

4 Classical Methods

As a baseline comparison to the use of quantum computers to solve QUBO problems, we also investigated two classical optimization techniques.

Gurobi is a suite of optimization solvers compatible with multiple programming languages, capable of solving both linear and quadratic problems. The software automatically selects algorithms from its toolkit, including simplex and parallel barrier methods, based on problem characteristics. For further details, see Ref. [24]. In this work, we used Gurobi to solve our QUBO problem, which is a special case of quadratic optimization. Gurobi has been used as a classical benchmark for solving QUBO problems in comparative studies with quantum and digital annealers [25,26].

Simulated Annealing [27] (SA) is a standard approach to finding the optimum of a function that has many local minima in a large search space. The algorithm is especially effective for discrete space exploration. It navigates the solution space through controlled random fluctuations, where a temperature-dependent probability function governs state transitions. SA has been used to find solutions to various NP-complete combinatorial problems such as the Traveling Salesman Problem [28], Minimum Linear Arrangement [29], and instances of the Packing Problem [30].

5 Variational Quantum Eigensolver

The VQE is a hybrid quantum-classical algorithm for finding the lowest eigenvalue of a Hamiltonian that is in the form of a Pauli string [31]. For further details about the VQE, see Ref. [32]. The VQE has become a pivotal algorithm for the NISQ era. The approach employs a parameterized quantum circuit to compute the Hamiltonian expectation value with respect to parameters θ, coupled with a classical optimization routine that minimizes this value through iterative θ updates.

The VQE uses the inequality

$$E^{H}_{min} \leq \langle 0|U^{\dagger}(\theta)HU(\theta)|0\rangle, \tag{11}$$

and the variational principle to find a tight upper bound on the lowest eigenvalue, E^{H}_{min}, of a Pauli string matrix, H, which is defined as

$$H = \sum_{\gamma} h_{\gamma}P_{\gamma} \tag{12}$$

$$P_{\gamma} = \bigotimes_{i=0}^{q} \sigma_{m_i}, \tag{13}$$

where $m_i \in \{0, 3\}$ tells us the Pauli matrix and q is the number of qubits. The two Pauli matrices that we make use of in this problem formulation are:

$$\sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2, \tag{14}$$

and

$$\sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{15}$$

The parameter values are then chosen by a classical optimization routine to find the value for

$$\min_{\theta} \langle 0|U^{\dagger}(\theta)HU(\theta)|0\rangle \tag{16}$$

There are two main features of the VQE that we must design and control for efficient implementation. These are: the design of the parameterized circuit (the *ansatz*) and the choice of optimization routine. Our choice of ansatz must create states that have sufficient overlap with the optimal state in the solution space. When using the VQE to solve QUBO problems, this is fairly trivial, as one parameter on each qubit would be sufficient (we are only looking for basis states). However, selecting an appropriate optimization routine presents a non-trivial challenge that critically impacts algorithm performance. In this work, we examine the effectiveness of three different optimization routines: Powell's Optimization (PO), Constrained Optimization BY Linear Approximation (COBYLA), and Bayesian Optimization (BO).

Powell optimization

PO is a widely known standard optimization routine for finding the global minimum of a function without the use of derivatives. It works well for the VQE (in small parameter spaces) but requires the system to use a high shot[1] count in order not to be too greatly affected by the quantum measurement noise. While

[1]Number of calls of the quantum circuit to gain one expected value evaluation.

this method is robust with high iterations for a black-box problem and does not use derivative information, it is poorly suited to the quantum formulation due to the sheer number of shots and iterations required.

For this work, we make use of `scipy.optimize.minimize`'s option `'powell'`.

COBYLA

COBYLA is a trust region-based, surrogate-assisted method for finding the global minimum of a function. Similar to PO, COBYLA requires sufficient shots in order not to be too greatly affected by the noise. However, it is more resilient than PO. COBYLA has been used for the VQE several times in the literature. Liu et al. [33] used it as a test for their Layer-VQE (L-VQE) approach to generating ansatz, and Tim Schwägerl et al. [34] used it to minimize a QUBO problem for reconstructing particle track results from a Large Hadron Collider (LHC). COBYLA has shown good promise for this black-box optimization. The lack of need for derivative information and the use of surrogate models make COBYLA fairly well-suited for the noisy environment in which we are working. Its trust-region approach provides some inherent noise robustness. COBYLA is efficiently implemented in Python, which makes it a useful choice.

For this work, we make use of `scipy.optimize.minimize`'s option `'COBYLA'`.

Bayesian Optimization

BO is a surrogate-assisted method for finding the global minimum of a function, which models the underlying function as a Gaussian Process Regression (GPR) and then updates this model based on samples, according to Bayes' theorem [35,36]. BO has been extensively studied in the field of hyperparameter optimization for machine learning (e.g., Refs. [37–39]). BO is well suited to the noisy environment of quantum optimization, as it takes this variance into account when building surrogate Gaussian Process (GP) models. BO has a high computational cost, especially as the number of iterations grows.

For this work, we use a BO that we have coded. The definitions that we use are the same as presented in [40], except for the choice of *acquisition function*. We use the periodic kernel, for K parameters, defined as,

$$k^P(\theta^1, \theta^2) = \sigma^2 \prod_{i=1}^{K} \exp\left[\frac{-2}{l^2}\sin^2\left(\pi\left|\frac{\theta_i^1 - \theta_i^2}{p}\right|^2\right)\right], \tag{17}$$

where σ is the expected variance of the GP, l is the length scale over which we would expect data points to be correlated, and p is the period of the underlying function.

We make use of Expected Improvement (EI) as the acquisition function. If we define our best guess so far to be $E_{min} = \min(E_1, ..., E_n)$, EI is defined as

$$a_{EI}(\theta) = \mathbb{E}_\theta[\max(0, E_{min} - E(\theta))] \tag{18}$$

where $\mathbb{E}_\theta[\cdot]$ is evaluated over all the possible $E(\theta)$ from the surrogate model.

Expected Improvement can be used in its explicit closed form, which is,

$$\begin{aligned} a_{EI}(\theta) = {} & (E_{min} - E(\theta))\, \Phi\left(\frac{E_{min} - E(\theta)}{\Delta E(\theta)}\right) \\ & + \Delta E(\theta)\phi\left(\frac{E_{min} - E(\theta)}{\Delta E(\theta)}\right). \end{aligned} \tag{19}$$

where Φ is the Normal cumulative distribution function and ϕ is the corresponding probability density function. We find the minimum of a_{EI} using COBYLA.

5.1 Mapping QUBO to the VQE

We can use the VQE to solve QUBO problems using the transformation

$$x^i \mapsto \frac{1}{2}\left(\sigma_0^i + \sigma_3^i\right), \tag{20}$$

where σ_3^i denotes the Pauli-Z spin matrix acting on the i^{th} qubit. We then find the groundstate eigenpair of the Hamiltonian

$$H = \sum_{i,j=1}^{q} Q_{ij}\frac{1}{4}\left((\sigma_0^i + \sigma_3^i)(\sigma_0^j + \sigma_3^j)\right). \tag{21}$$

Once found, we take the parameters that produce this groundstate, measure the state in the $\sigma_3^{\otimes q}$ basis, and the state with the highest likelihood of being measured, which also meets our constraints, is our solution.

Since our Hamiltonians only consist of σ_0 and σ_3 terms, they are diagonal; hence, we only need one parameter per qubit to map a sufficient amount of the space to receive the optimum. However, to aid the classical optimizer in the noisy space of quantum measurements, q layers of rotation and entangling Controlled-X or Controlled-NOT ($CNOT$) gates will be used.

We note here that due to the density of the weight matrix Q defined in Eq. (10), the Hamiltonian is fully dense, containing q^2 nonzero terms. This means that this is a worst-case scenario in terms of stress on the VQE.

5.2 Dimensionallity Expressivity Analysis

In variational quantum circuits, Dimensionality Expressivity Analysis (DEA) [41] serves as a key tool for assessing independence among quantum gate parameters. This method identifies whether gate parameters represent independent degrees of freedom or exhibit redundancy. Full details on DEA methodology are available in [42].

DEA is more applicable when looking at complex systems, i.e., when we are not only searching for basis states, but a specific subspace is required.

When performing this analysis on our circuit, we observe that all parameters remain non-redundant provided none are initialized at $(2k+1)\pi/4$ or $2k\pi/4$ for $k \in \{0,1\}$. To guarantee this condition, we initialize parameters by sampling uniformly from $[0, 2\pi)$ using `numpy.random.uniform`.

Further details about DEA are included in Appendix A.

5.3 Conditional Value at Risk

A significant advancement in applying VQE to combinatorial optimization involves adopting the Conditional Value at Risk (CVaR) measure from financial mathematics. While Ref. [43] implemented CVaR-VQE using the sample mean, our work applies this approach to the quantum expectation value. See Ref. [43] for complete methodology and advantages of using CVaR.

For a Hamiltonian on which we carry out K measurements (shots), our objective function terms are,

$$\text{VQE} \to \sum_{\gamma=1}^{K} H_\gamma \tag{22}$$

If we now order the measurements so that $H_k \leq H_{k+1}$, the CVaR adaptation is to introduce a new parameter: $0 < \alpha \leq 1$ such that,

$$\text{CVaR-VQE} \rightarrow \sum_{\gamma=1}^{\lceil \alpha K \rceil} H_\gamma \tag{23}$$

This approach truncates measurements to include only the terms contributing most significantly to the minimization objective, specifically, the largest negative contributions. The CVaR method considers only the lowest αK measurements, where adjusting α enables faster convergence to the minimal energy state. However, very small α values may yield unstable results due to severe measurement reduction. The standard VQE is recovered when $\alpha = 1.0$.

Note that the minimum value of CVaR-VQE is not necessarily equal to the true minimum of the VQE, but the correct solution state can be the same.

6 Design of the Tests of the Algorithms

We evaluated power production across methods using $l_{\text{grid}} = 4$ grids, limited by computational constraints. Through exhaustive search, we determined the maximum achievable power for this grid size. Given the WFLO problem's numerous local optima, each algorithm was executed 36 times with distinct initial conditions. To assess scaling behavior, quantum simulators were tested for $l_{\text{grid}} \in \{2, 3, 4\}$ while classical optimizers were extended to $l_{\text{grid}} \in \{3, \dots, 10\}$.

Justification for 36 being a sufficient sample size can be found in Appendix B.

6.1 Test Model

To test these different algorithms, we will use the following parameter values (the Mosetti benchmark case is from [5]),

$$D = \text{Mosetti second benchmark case}$$
$$x = 1, \quad r = 1.5, \quad m = 4, \quad \xi = 0.$$

We note that these parameters are chosen for mathematical simplicity in this initial test work and do not relate to real turbines. As a result, our toy model power output should be seen as a form of cost. m is the maximum number of turbines allowed, and ξ is the minimum distance allowed between turbines. In this work, we have set ξ to zero, meaning turbines may be on adjacent sites. Quantum simulation constraints limit our study to an $l_{\text{grid}} = 4$ case (16 turbine positions), which directly corresponds to a 16-qubit system requirement.

Mosetti's second benchmark case is defined as:

$$D = \{\{10k, 12ms^{-1}, 1/36\}\}_{k=0}^{36} \tag{24}$$
$$= \{\{0, 12ms^{-1}, 0.028\}, \dots, \{350, 12ms^{-1}, 0.028\}\} \tag{25}$$

6.2 Computational Details of Simulations

The quantum simulations were conducted using IBM's `qiskit:0.39.2` package for Python [12].

Both PO and COBYLA were utilized via `scipy.optimize.minimize` - employing the methods `'powell'` and `'COBYLA'` respectively as part of the `scipy` library [44]. BO was coded up for this work, using a heavily modified version of the code used in [45], where we no longer use `sklearn.gaussian_process.GaussianProcess-Regressor`. This had problems when using the periodic kernel, and so a self-made algorithm was required.

SA was implemented using the package `pyqubo` [46]. This library has a built-in annealer: `neal.SimulatedAnnealingSampler()`. Once the system has been turned into a QUBO problem via `model.to_qubo()`, `pyqubo` has a wrapper for the D-Wave Ocean Software Development Kit (SDK), enabling seamless execution on D-Wave systems.

Gurobi was implemented using `gurobipy` [24], with an academic license obtained via the University of Plymouth.

As well as the previously mentioned methods, we also investigated `qiskit`'s built-in VQE QUBO solver. This uses their `SamplingVQE` function, and then performs a `MinimumEigenOptimizer` over it to find the groundstate. The ansatz used is the same as the one described in Section 5.1. This method acts as a control, as it is the VQE method with no statistical noise present. We use COBYLA as the classical optimization routine for this method. Details about this method can be found in Ref. [47].

All of the simulations for the results provided below were run using the computational facilities of the High Performance Computing Centre located at the University of Plymouth. The Central Processing Units (CPUs) were dual-node Intel Xeon E5-2683v4, each with 16 cores, at the time of simulation. Task farming was used to run the simulations using all the cores.

7 Results

7.1 How Solutions Are Selected

For both the Gurobi solver and simulated annealing, the selection of the solution is trivial; it will be the final binary string found by the algorithm.

However, for the VQE, this is less obvious. Once the optimization process has finished, we are given a set of parameters that correspond to the quantum circuit gates. We then input these values back into the circuit and measure in the σ_3 basis on all qubits; this produces a distribution over all possible basis states. From this distribution, we then take the state which has the highest probability and only has $m|1\rangle$ terms. We only accepted solutions that have exactly the correct number of turbines, which is four for our test case.

7.2 Discussion about the Degeneracy of Solutions

As for many other QUBO problems, there are many degenerate optimal solutions. For the case $l_{grid} = 4$, there are 79 unique optimal solutions out of 1820 binary vectors that meet the constraints and 65,536 total possible binary strings of length 16. A list of all optimal solutions can be found in Appendix C. Each of these optimal solutions has a power output of 2304.0 kW.

Some of these degeneracies are due to the fact that there are optimal solutions for the conditions in smaller system sizes. The smaller system solutions can then be embedded in the larger systems. No solutions exist with output 2304.0 kW for $l_{grid} \leq 2$.

An alternative approach to analyzing optimal solution structures involves examining average turbine placements. For a given solution X_*^i, represented as a binary matrix encoding turbine positions on the grid, we compute the mean location across all optimal configurations. We map from solutions to grids by using the labelling as in Fig. 3. We then calculate the average placement as:

$$\langle X \rangle = \frac{1}{\#\text{solutions}} \sum_i X_*^i \tag{26}$$

The results from this can be seen in Fig. 8. Note here that the sum of these should equal four, as this is the maximum number of turbines; some rounding errors may occur due to limited precision in the printed values. From this, we see that there are more optimal solutions with turbines in the outer corners of the grid. This makes sense physically with the wind regime used, as it maximizes the distance between any two turbines and thus minimizes any possible wake effects.

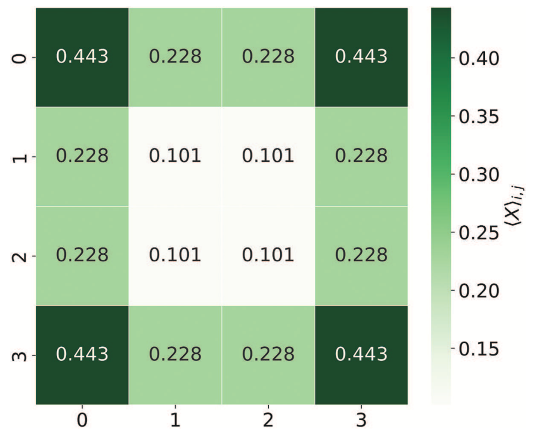

Figure 8: Heatmap of $\langle X \rangle$ values for the degenerate solutions with maximum power. All the values sum to 4 (the number of turbines)

7.3 Discussion of the Results

Our simulations showed that PO achieved reasonable computation times for $l_{\mathrm{grid}} \in \{2, 3\}$, but failed to converge for $l_{\mathrm{grid}} = 4$ due to excessive shot counts and function evaluations. Consequently, we present results only for the surrogate-based methods (COBYLA and BO) and the two classical algorithms.

Fig. 9 reports the results for the power output from the different algorithms investigated for the $l_{grid} = 4$ case. Each method was run 36 times; the box plots show the spread of these results. The x-axis shows the names of the different methods, and the number under the name of the VQE method is the CVaR α value.

From Fig. 9 we can draw our conclusions. The Gurobi method consistently reaches the optimal solution, outperforming all other methods. This is also true for the noiseless COBYLA-VQE method from `qiskit`. The VQE variants yield comparable solution quality both among themselves and relative to SA.

We see that each of the COBYLA-CVaR methods is capable of achieving the optimal solution. We also observe that using $\alpha < 1$ has minimal impact on the quality of results. We see that the Bayesian-CVaR has very comparable results.

Table 1 shows the average power output of each of the methods as a percentage of the optimal power output. We see that each of these methods performs very well on average, with the lowest average output being 93.3% of the maximum.

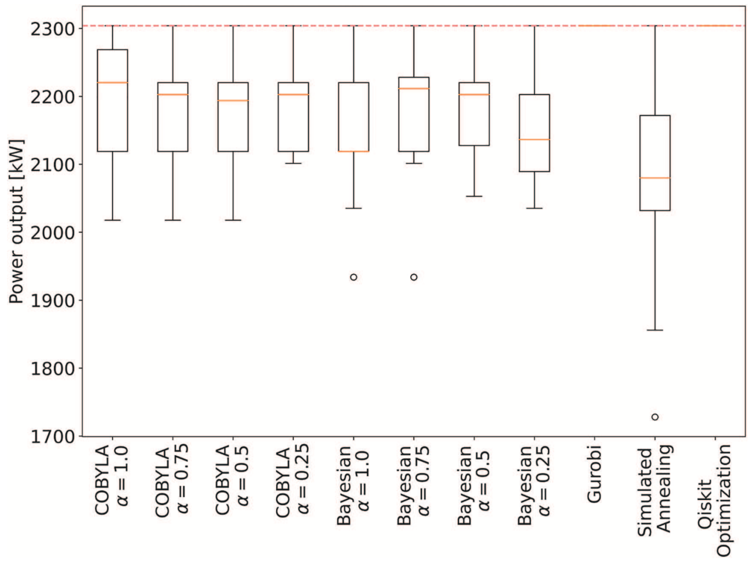

Figure 9: Each value along the x-axis represents the different methods. The box plots show the power obtained from running the problems 36 times. The red dashed line is the optimal solution power output, found through an exhaustive list search. The Gurobi method and Qiskit Optimization may look empty: the optimal solution was found every time. The circle points that are not within the boxes are outlier points

Table 1: Average output power from solutions of different VQE-based methods as a percentage of the optimal power output

Method	Average power output (% of optimal)
COBYLA-1.00	95.5
COBYLA-0.75	94.7
COBYLA-0.50	94.5
COBYLA-0.25	95.2
Bayes-1.00	93.3
Bayes-0.75	94.2
Bayes-0.50	94.6
Bayes-0.25	93.4

7.4 Timing the Results

Beyond solution quality, we assess computational efficiency by measuring algorithm runtime. Using Python's `time` module, we recorded end-to-end execution times, with complete results presented in Table 2.

Table 2: Average time in seconds taken for the different optimization algorithms

l_{grid}	Gurobi (s)	Simulated annealing (s)	COBYLA-CVaR (s)	BO-CVaR (s)
2	–	–	19.74	250.11
3	0.09	0.04	561.78	5741.43
4	1.11	0.85	11546.10	55454.77
5	3.91	3.31	–	–

(Continued)

Table 2 (continued)

l_{grid}	Gurobi (s)	Simulated annealing (s)	COBYLA-CVaR (s)	BO-CVaR (s)
6	14.51	11.49	–	–
7	107.11	106.62	–	–
8	374.72	377.49	–	–
9	983.27	985.57	–	–
10	2240.78	2220.44	–	–

Note: All timings are averaged results. Missing entries (–) indicate the method was not evaluated at this grid size.

These results demonstrate that the quantum simulator exhibits significantly longer runtimes than classical algorithms at fixed system sizes. More importantly, analyzing how runtime scales with increasing system size reveals each algorithm's empirical computational complexity, which gives us a good metric as quantum resources grow in size. We can study this by looking at the regression coefficients of LogLog plots of the timings.

The time to solve the QUBO problem is expected to depend on the system size exponentially. While the worst-case complexity is exponential, our results show power-law scaling, due to the simplicity of the scaling problem and limited data. As we only have three data points, we parameterize the time to solution as a polynomial with exponent α as

$$\text{Time to solve} \propto (V)^{\alpha},$$

where $V = l_{grid}^2$ is the system volume. Fig. 10 shows the Log of time taken to the solution vs. the Log of the system volume. The power-law fit coefficients can be found in Table 3. The results from all four algorithms in Fig. 10 have similar slopes.

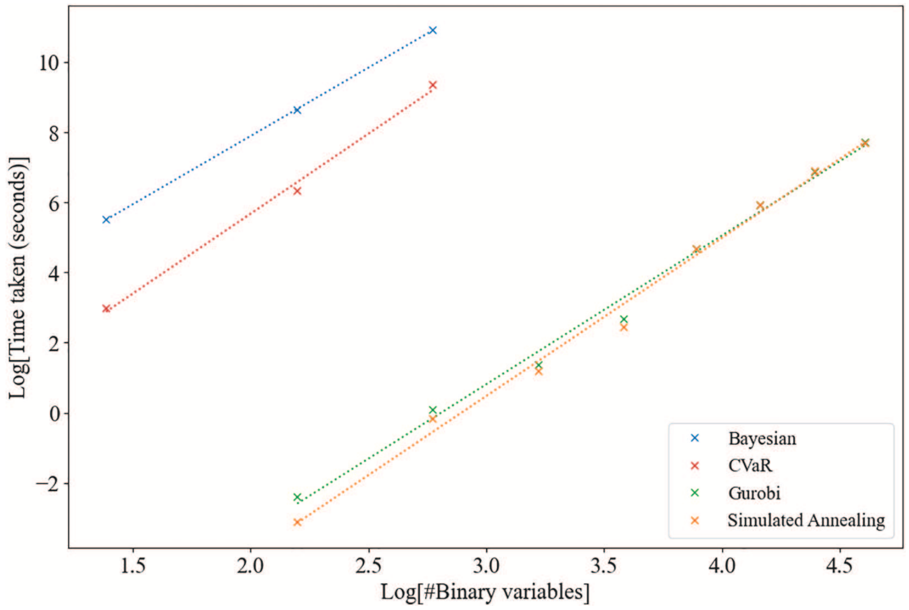

Figure 10: LogLog plots of the time taken for the different methods over different system sizes

Table 3: Slope and intercept from linear regression of Log (time) vs. Log (number of binary variables) for the different algorithms

Method	Gradient	Intercept
Gurobi	4.24	−5.18
Simulated Annealing	4.51	−5.67
CVaR-VQE using COBYLA	4.57	−1.50
CVaR-VQE using BO	3.89	0.04

We find that: Gurobi ~ $O(V^{4.24})$, SA ~ $O(V^{4.51})$, COBYLA ~ $O(V^{4.57})$ and Bayesian ~ $O(V^{3.89})$. The excellent LogLog linear fits support this approximation for our data range. This result is particularly noteworthy because while Gurobi and SA represent actual algorithm implementations, the quantum approach runs on a classical simulator. We anticipate that true quantum hardware would significantly reduce these computation times.

We note that BO, run on the simulator, scales better than all the other methods, having the lowest polynomial scaling; this does come with the caveat that the pre-factor is much larger than the other systems. We can look at this more closely by searching for the intersection point of the Bayesian and Gurobi timing lines, shown in Fig. 11. We can estimate that the simulated Bayesian-CVaR method would be faster than the Gurobi method for a system volume of $\approx 10^{15}$, which is a square grid, with $l_{\text{grid}} \approx 31,622,776$. This scaling extrapolation shows that even when simulating, there may be a point at which this algorithm is faster. However, more data at larger scales is required to make a concrete claim, to increase the accuracy of our power-law scaling approximation. We emphasize that these scaling laws are empirical observations, not theoretical complexity predictions.

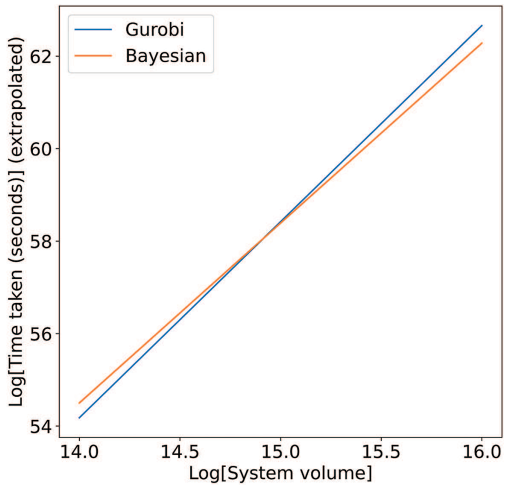

Figure 11: Extrapolation of time taken for Gurobi and Bayesian Optimization to find the point at which Bayesian would be faster

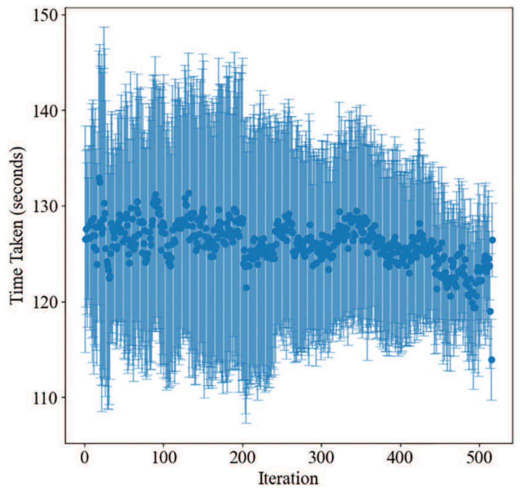

Figure 12: Time taken per iteration when using COBYLA applied to our WFLO problem. Each iteration takes the same amount of time as the previous

Another important consideration when comparing COBYLA and BO in this setting is the time taken per iteration. Figs. 12 and 13 show the iteration timings for runs of our WFLO problem with $l_{\text{grid}} = 4$ (each of the data points shows samples from 64 runs, where the points are the average and the error bars are the standard deviation of these samples), for COBYLA and BO, respectively. For COBYLA, the time per iteration remains approximately constant across the optimization, indicating no significant scaling. In contrast, for BO, the time per iteration increases, appearing to scale exponentially, as highlighted in the LogLog plot in Fig. 14. Despite this exponential growth in time per iteration, the overall time to solution is *better*, as the problem scales, compared to COBYLA, as demonstrated in Fig. 10.

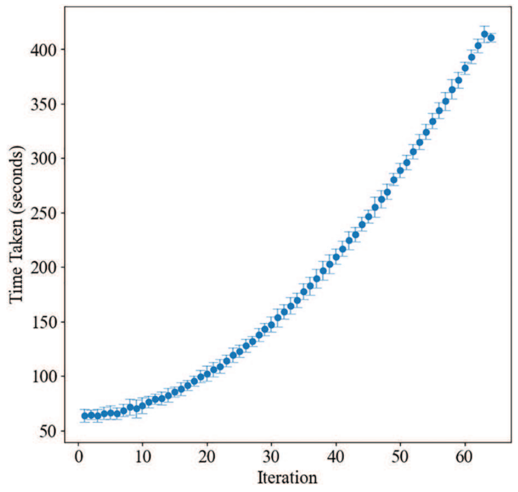

Figure 13: Time taken per iteration when using BO applied to our WFLO problem. The time taken seems to scale exponentially

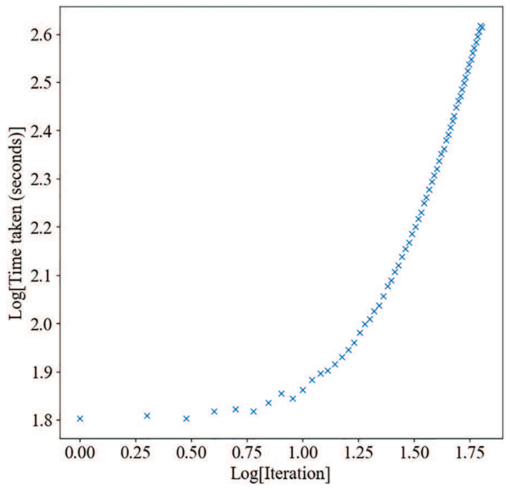

Figure 14: LogLog plot of time taken per iteration when using BO applied to our WFLO problem. The time taken seems to scale exponentially

7.5 Conclusions

We successfully demonstrated the novel application of quantum circuit methods to the WFLO problem, establishing this as a promising new approach for finding optimal turbine configurations. Our quantum-classical hybrid method shows particular potential for future quantum hardware implementations. The VQE-based methods demonstrate viability for identifying optimal or near-optimal solutions while exhibiting favorable time complexity scaling, provided a sufficient number of measurements have been performed. The Gurobi optimizer always finds the optimal solution, outperforming the noisy VQE-based method. This is not surprising, as Gurobi has been developed to specifically solve linear and quadratic programming problems, of which QUBO is a special case.

We have only investigated small system sizes due to computational constraints in the simulator. Numerous benchmarking studies have evaluated classical algorithms for WFLO problems using high-resolution grids, typically at least 100 points. This would require 100 qubits in our VQE-based approach, and possibly many more when error correction is used.

Interestingly, we observe comparable scaling exponents in runtime vs. system size between Gurobi and the simulated VQE approach. From our results, the performance of the VQE method critically depends on the performance of the optimizer that runs on the classical computer.

The development of the VQE is an active area of research. For example, Ref. [48] has studied the iteration complexity of Variational Quantum Algorithms in a noisy environment. A generalization of CVaR-VQE has been proposed, called the Filter-VQE (F-VQE). F-VQE uses a technique based on filtering operators to achieve faster and more reliable convergence to the optimal solution [49]. When dealing with high gate count variational circuits, exponentially vanishing gradients, or *barren plateaus* [50], have been found. When scaling to the system sizes discussed in Section 7.4, barren plateaus will become increasingly prevalent due to the growing number of circuit parameters. Recent work [51] addresses this challenge through parallel optimization routines (particles) that dynamically relocate when encountering vanishing gradients or noise-dominated measurements.

We are encouraged by the results for the solution of the WFLO obtained on the quantum computing simulator. Next, we will study the resilience of the implemented methods against quantum errors on the simulators, before using a quantum computer. It will be interesting to compare the performance on circuit-based quantum computers to that of adiabatic quantum systems.

Ultimately, as quantum computers become larger and more reliable, we hope that the methods we have begun to develop here will help wind farms produce more power, accelerating the transition to Net Zero.

Acknowledgement: We thank Julian Stander for the discussions about Bayesian optimization. The simulations for this work were carried out using the computational facilities of the High Performance Computing Centre located at the University of Plymouth–https://www.plymouth.ac.uk/about-us/university-structure/faculties/science-engineering/hpc (accessed on 20 July 2025).

Funding Statement: The authors received no specific funding for this study.

Author Contributions: James Hancock: Writing—review & editing, Writing—original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. Matthew Craven: Writing—review & editing, Writing—original draft, Visualization, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. Craig McNeile: Writing—review & editing, Visualization, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization. Davide Vadacchino: Writing—review & editing, Supervision, Project administration, Methodology, Investigation, Conceptualization. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and simulation code that support the findings of this study are openly available at GitHub: https://github.com/jamesshancock/WindfarmLayoutOptimization (accessed on 20 July 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Appendix A Dimensional Expressivity Analysis

Circuit expressivity analysis provides a systematic approach to evaluate and enhance system performance by quantifying the parameterized quantum circuit's capability to explore the solution space. For an objective function $F\left(\vec{\theta}\right)$, we can then perturb θ_k by $\delta\theta_k$ to produce $F\left(\vec{\theta} + \hat{e}_k\delta\theta_k\right) = f$. If we can also have $F\left(\vec{\theta} + \sum_{i\neq k}\hat{e}_i\delta\theta_i\right) = f$, θ_k is redundant. To perform this analysis, we are using the method described by Lena Funcke et al. in [42]. The method is as follows:

1. This can be checked by considering the real partial Jacobians J_k of C (C is the matrix of the circuit applied to the state $|0\rangle^{\otimes q}$):

$$J_k(\theta) = \begin{pmatrix} Re(\partial_1 C)...Re(\partial_k C) \\ Im(\partial_1 C)...Im(\partial_k C) \end{pmatrix}$$

Here it is key to understand that $\partial_k C$ is itself a $2^q \times 1$ vector, and thus J_k is a $2^{q+1} \times k$ matrix, where we start with $k = 1$ for the first parameter.

2. We then check the rank of the matrix J_k for each k, and if adding a new parameter does not increase the rank, the parameter must be redundant.

To efficiently check the rank of J_k, we consider the matrix $S_k = J_k^* J_k$. Thus, if we check that S_k is invertible, we know that all the parameters are independent (i.e., not redundant).

Appendix B Justification for Thirty-Six Samples

When gathering results for this work, we collected 36 samples per optimization algorithm. Here, we focus specifically on COBYLA with CVaR ($\alpha = 0.25$), analyzing an extended sample set (284 runs) to validate our initial findings. The statistical similarity between the larger sample and our original 36-run dataset confirms the adequacy of the smaller sample size for drawing robust conclusions.

The initial 36 samples taken were:

```
2136.54, 2118.96, 2202.69, 2220.27,
2220.27, 2220.27, 2268.84,
2220.27, 2286.42, 2220.27, 2136.54,
2118.96, 2118.96, 2220.27,
2101.38, 2202.69, 2220.27, 2286.42,
2202.69, 2286.42, 2304.0,
2118.96, 2286.42, 2118.96, 2118.96,
2136.54, 2136.54, 2268.84,
2118.96, 2118.96, 2101.38, 2286.42,
2220.27, 2286.42, 2202.69,
2118.96
```

The extra 284 samples taken were:

```
2251.25, 2202.69, 2185.11, 2185.11,
2136.54, 2136.54, 2286.42,
2118.96, 2268.84, 2304.0, 2202.69,
2136.54, 2220.27, 2268.84,
2101.38, 2118.96, 2268.84, 2220.27,
2220.27, 2202.69, 2202.69,
1933.92, 2118.96, 2202.69, 2268.84,
2220.27, 2220.27, 2220.27,
2268.84, 2220.27, 2185.11, 2136.54,
2118.96, 2202.69, 2251.25,
2304.0, 2052.82, 2118.96, 2286.42,
2136.54, 2118.96, 2220.27,
2118.96, 2118.96, 2268.84, 2304.0,
2101.38, 2268.84, 2286.42,
2035.23, 2304.0, 2202.69, 2118.96,
2304.0, 2202.69, 2017.65,
2268.84, 2220.27, 2136.54, 2304.0,
2202.69, 2118.96, 2268.84,
2136.54, 2136.54, 2202.69, 2118.96,
2035.23, 2202.69, 2304.0,
2118.96, 2220.27, 2136.54, 2185.11,
2118.96, 2220.27, 2251.25,
2220.27, 2304.0, 2202.69, 2202.69,
2220.27, 2268.84, 2304.0,
2185.11, 2136.54, 2202.69, 2118.96,
2185.11, 2220.27, 2118.96,
2118.96, 2268.84, 2220.27, 2185.11,
2136.54, 2251.25, 2304.0,
2268.84, 2202.69, 2220.27, 2185.11,
2202.69, 2118.96, 2220.27,
2220.27, 2220.27, 2286.42, 2268.84,
2220.27, 2185.11, 2202.69,
2101.38, 2202.69, 2220.27, 2202.69,
2304.0, 2304.0, 2220.27,
2202.69, 2220.27, 2286.42, 2220.27,
2118.96, 2118.96, 2220.27,
2202.69, 2035.23, 2118.96, 2202.69,
2286.42, 2118.96, 2101.38,
2118.96, 2202.69, 2304.0, 2136.54,
2304.0, 2185.11, 2202.69,
2286.42, 2101.38, 2202.69, 2220.27,
2220.27, 2220.27, 2220.27,
2220.27, 2136.54, 2220.27, 2101.38,
2220.27, 2286.42, 2118.96,
2220.27, 2220.27, 2286.42, 2220.27,
2136.54, 2136.54, 2268.84,
2101.38, 2202.69, 2101.38, 2035.23,
2220.27, 2136.54, 2202.69,
2136.54, 2286.42, 2185.11, 2286.42,
2220.27, 2118.96, 2118.96,
2220.27, 2202.69, 2202.69, 2118.96,
2118.96, 2118.96, 2202.69,
2220.27, 2202.69, 2220.27, 2202.69,
2286.42, 2118.96, 2202.69,
2136.54, 2220.27, 2268.84, 2118.96,
2251.25, 2286.42, 2136.54,
2136.54, 2286.42, 2286.42, 2220.27,
2202.69, 2286.42, 2136.54,
2220.27, 2202.69, 2220.27, 2118.96,
2202.69, 2304.0, 2220.27,
2202.69, 2304.0, 2286.42, 2304.0,
2286.42, 2101.38, 2220.27,
2101.38, 2220.27, 2202.69, 2202.69,
2220.27, 2035.23, 2304.0,
2304.0, 2202.69, 2118.96, 2185.11,
2202.69, 2220.27, 2118.96,
2101.38, 2202.69, 2268.84, 2304.0,
2202.69, 2017.65, 2118.96,
2286.42, 1933.92, 2304.0, 2118.96,
2202.69, 2286.42, 2220.27,
2136.54, 2136.54, 2185.11, 2017.65,
2185.11, 2220.27, 2202.69,
2118.96, 2136.54, 2118.96, 2220.27,
2017.65, 2251.25, 2136.54,
2202.69, 2035.23, 1933.92, 2136.54,
2017.65, 2220.27, 2118.96,
2101.38, 2220.27, 2286.42, 2118.96,
2286.42, 2220.27, 2286.42,
2220.27, 2286.42, 2202.69, 2268.84,
2101.38, 2251.25, 2220.27,
2035.23, 2286.42, 2202.69, 2202.69
```

The sample mean of the 36 initial samples is: 2193.1, and the sample mean of the extra is: 2192.4. We can see that these are similar values (0.03% difference), and so we can conclude the method's effectiveness *on average* by only taking 36 samples. This is very useful as minimizing time and resource waste is an important factor when carrying out simulations.

Appendix C Degeneracy of Solutions

Below is a list of all possible optimal solutions for the problem defined in Section 6.1.

```
(0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
(0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0),
(0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
(0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0),
(0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0),
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0),
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0),
(0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1),
(0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1),
(0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1),
(0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1),
(0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0),
(0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0),
(1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1),
(1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1),
(1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0),
(1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1),
(1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1),
(1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0),
(1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1),
(1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0),
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0),
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0),
(1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1),
(1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0),
(1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1),
(1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0),
(1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0),
(1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1),
(1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1),
(1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0),
(1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0)
```

References

1. Veers P, Dykes K, Lantz E, Barth S, Bottasso CL, Carlson O, et al. Grand challenges in the science of wind energy. Science. 2019;366(6464):eaau2027. doi:10.1126/science.aau2027.
2. Giani A, Eldredge Z. Quantum computing opportunities in renewable energy. Nat Comput Sci. 2021;2(5):393. doi:10.1038/s43588-021-00032-z.
3. Ajagekar A, You F. Quantum computing and quantum artificial intelligence for renewable and sustainable energy: a emerging prospect towards climate neutrality. Renew Sustain Energ Rev. 2022;165(6):112493. doi:10.1016/j.rser.2022.112493.
4. Abbas A, Ambainis A, Augustino B, Bärtschi A, Buhrman H, Coffrin C, et al. Quantum optimization: potential, challenges, and the path forward. arXiv:2312.02279. 2023. doi:10.2172/2229681.
5. Mosetti G, Poloni C, Diviacco B. Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. J Wind Eng Indust Aerodynam. 1994;51(1):105–16. doi:10.1016/0167-6105(94)90080-9.
6. Manikowski PL, Walker DJ, Craven MJ. Multi-objective optimisation of the benchmark wind farm layout problem. J Mar Sci Eng. 2021;9(12):1376. doi:10.3390/jmse9121376.
7. Senderovich A, Zhang J, Cohen E, Beck JC. Exploiting hardware and software advances for quadratic models of wind farm layout optimization. IEEE Access. 2022;10:78044–55. doi:10.1109/ACCESS.2022.3193143.
8. Rajak A, Suzuki S, Dutta A, Chakrabarti BK. Quantum annealing: an overview. Philosoph Transact Royal Soc A Mathem Phys Eng Sci. 2023;381(2241):20210417. doi:10.1098/rsta.2021.0417.
9. McGeoch C, Farré P. Advantage processor overview. D-Wave Systems; 2022. [cited 2025 Jul 20]. Available from: https://www.dwavequantum.com/resources/white-paper/the-d-wave-advantage-system-an-overview/.
10. McGeoch C, Farré P, Boothby K. D-WAVE Advantage2 prototype. D-Wave Systems; 2022. [cited 2025 Jul 20]. Available from: https://www.dwavequantum.com/media/eixhdtpa/14-1063a-a_the_d-wave_advantage2_prototype-4.pdf.
11. Matsubara S, Takatsu M, Miyazawa T, Shibasaki T, Watanabe Y, Takemoto K, et al. Digital annealer for high-speed solving of combinatorial optimization problems and its applications. In: Proceeding of the 25th Asia South Pacific Design Automation Conference (ASP-DAC). Beijing, China: IEEE; 2020. p. 667–72. doi:10.1109/ASP-DAC47756.2020.9045100.
12. Qiskit contributors. Qiskit: an open-source framework for quantum computing. 2023. doi:10.5281/zenodo.2573505.
13. Preskill J. Quantum computing in the NISQ era and beyond. Quantum. 2018;2:79. doi:10.22331/q-2018-08-06-79.
14. Abbas A, Ambainis A, Augustino B, Bärtschi A, Buhrman H, Coffrin C, et al. Challenges and opportunities in quantum optimization. Nat Rev Phys. 2024;6(12):718–35. doi:10.1038/s42254-024-00770-9.
15. Zahedinejad E, Zaribafiyan A. Combinatorial optimization on gate model quantum computers: a survey. arXiv:1708.05294. 2017. doi:10.48550/arXiv.1708.05294.
16. Albash T, Lidar DA. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. Phys Rev X. 2018;8(3):31016. doi:10.1103/PhysRevX.8.031016.
17. Pearson A, Mishra A, Hen I, Lidar DA. Analog errors in quantum annealing: doom and hope. NPJ Quantum Inf. 2019;5(1):107. doi:10.1038/s41534-019-0210-7.
18. Yaacoby R, Schaar N, Kellerhals L, Raz O, Hermelin D, Pugatch R. Comparison between a quantum annealer and a classical approximation algorithm for computing the ground state of an Ising spin glass. Phys Rev E. 2022;105(3):035305. doi:10.1103/PhysRevE.105.035305.
19. Martin-Mayor V, Hen I. Unraveling quantum annealers using classical hardness. Sci Rep. 2015;5(1):15324. doi:10.1038/srep15324.
20. Zhang P, Romero D, Beck JC, Amon C. Solving wind farm layout optimization with mixed integer programming and constraint programming. In: Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. Berlin, Heidelberg: Springer Berlin/Heidelberg; 2013. p. 284–99. doi:10.1007/978-3-642-38171-3_19.

21. Donovan S. An improved mixed integer programming model for wind farm layout optimisation. In: Proceedings of the 41st Annual Conference of the Operational Research Society of New Zealand. Christchurch, New Zealand; 2006. p. 143–52.
22. Kochenberger G, Hao JK, Glover F, Lewis M, Lü Z, Wang H, et al. The unconstrained binary quadratic programming problem: a survey. J Combinat Optimiz. 2014;28(1):58–81. doi:10.1007/s10878-014-9734-0.
23. Khanali M, Ahmadzadegan S, Omid M, Keyhani Nasab F, Chau KW. Optimizing layout of wind farm turbines using genetic algorithms in tehran province. Iran Int J Energy Environ Eng. 2018;9(4):399–411. doi:10.1007/s40095-018-0280-x.
24. Gurobi Optimization LLC. Gurobi optimizer reference manual; 2024. [cited 2023 Jun 27]. Available from: https://docs.gurobi.com/projects/optimizer/en/current/index.html.
25. Şeker O, Tanoumand N, Bodur M. Digital Annealer for quadratic unconstrained binary optimization: a comparative performance analysis. Appl Soft Comput. 2022;127(1):109367. doi:10.1016/j.asoc.2022.109367.
26. Nakano K, Takafuji D, Ito Y, Yazane T, Yano J, Ozaki S, et al. Diverse adaptive bulk search: a framework for solving QUBO problems on multiple GPUs. In: 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). St. Petersburg, FL, USA: IEEE; 2023. p. 314–25. doi:10.1109/IPDPSW59300.2023.00060.
27. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–80. doi:10.1126/science.220.4598.671.
28. Wang Z, Geng X, Shao Z. An effective simulated annealing algorithm for solving the traveling salesman problem. J Computat Theorl Nanosci. 2009;6(7):1680–6. doi:10.1166/jctn.2009.1230.
29. Rodriguez-Tello E, Hao JK, Torres-Jimenez J. An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. Comput Operat Res. 2008;35(10):3331–46. doi:10.1016/j.cor.2007.03.001.
30. Tole K, Moqa R, Zheng J, He K. A simulated annealing approach for the circle bin packing problem with rectangular items. Comput Indust Eng. 2023;176(3):109004. doi:10.1016/j.cie.2023.109004.
31. Peruzzo A, McClean J, Shadbolt P, Yung MH, Zhou XQ, Love PJ, et al. A variational eigenvalue solver on a photonic quantum processor. Nat Commun. 2014;5(1):4213. doi:10.1038/ncomms5213.
32. Tilly J, Chen H, Cao S, Picozzi D, Setia K, Li Y, et al. The variational quantum eigensolver: a review of methods and best practices. Phys Rep. 2022;986(5):1–128. doi:10.1016/j.physrep.2022.08.003.
33. Liu X, Angone A, Shaydulin R, Safro I, Alexeev Y, Cincio L. Layer VQE: a variational approach for combinatorial optimization on noisy quantum computers. IEEE Trans Quantum Eng. 2022 Jan;3(3):1–20. doi:10.1109/TQE.2021.3140190.
34. Schwägerl T, Issever C, Jansen K, Khoo T, Kühn S, Tüysüz C, et al. Particle track reconstruction with noisy intermediate-scale quantum computers. arXiv:2303.13249. 2023. doi:10.48550/arXiv.2303.13249.
35. McClean JR, Romero J, Babbush R, Aspuru-Guzik A. The theory of variational hybrid quantum-classical algorithms. New J Phys. 2016;18(2):023023. doi:10.1088/1367-2630/18/2/023023.
36. Klco N, Dumitrescu EF, McCaskey AJ, Morris TD, Pooser RC, Sanz M, et al. Quantum-classical computation of Schwinger model dynamics using quantum computers. Phys Rev A. 2018 Sep;98(3):1014. doi:10.1103/PhysRevA.98.032331.
37. Snoek J, Larochelle H, Adams RP. Practical bayesian optimization of machine learning algorithms. In: Pereira F, Burges CJ, Bottou L, Weinberger KQ, editors. Advances in neural information processing systems. Vol. 25. Red Hook, NY, USA: Curran Associates, Inc.; 2012. doi:10.5555/2999325.2999464.
38. Nguyen V. Bayesian optimization for accelerating hyper-parameter tuning. In: 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). Sardinia, Italy: IEEE; 2019. p. 302–5. doi:10.1109/AIKE.2019.00060.
39. Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. Adv Neural Inf Process Syst. 2011;24. doi:10.5555/2986459.2986743.
40. Iannelli G, Jansen K. Noisy Bayesian optimization for variational quantum eigensolvers. In: The 38th International Symposium on Lattice Field Theory (LATTICE2021). Trieste, Italy; 2022. Vol. 396, 251 p. doi:10.22323/1.396.0251.
41. Funcke L, Hartung T, Jansen K, Kühn S, Stornati P. Dimensional expressivity analysis of parametric quantum circuits. Quantum. 2021;5:422. doi:10.22331/q-2021-03-29-422.

42. Hartung T, Funcke L, Jansen K, Kühn S, Schneider M, Stornati P. Dimensional expressivity analysis, best-approximation errors, and automated design of parametric quantum circuits. In: The 38th International Symposium on Lattice Field Theory (LATTICE2021); Trieste, Italy; 2022. 575 p. doi:10.22323/1.396.0575.
43. Barkoutsos PK, Nannicini G, Robert A, Tavernelli I, Woerner S. Improving variational quantum optimization using CVaR. Quantum. 2020 Apr;4:256. doi:10.22331/q-2020-04-20-256.
44. do Couto TG, Farias B, Diniz ACGC, de Morais MVG. Optimization of wind farm layout using genetic algorithm. E3S Web Conf. 2024;581:01024. doi:10.1051/e3sconf/202458101024.
45. Brownlee J. How to Implement Bayesian Optimization from Scratch in Python. [cited 2023 Jun 27]. Available from: https://machinelearningmastery.com/what-is-bayesian-optimization/.
46. Zaman M, Tanahashi K, Tanaka S. PyQUBO: python library for mapping combinatorial optimization problems to QUBO form. IEEE Transact Comput. 2022;71(4):838–50. doi:10.1109/TC.2021.3063618.
47. Qiskit Optimization Team. CVaR Optimization—Qiskit Optimization Tutorials; 2024. [cited 2025 May 8]. Available from: https://qiskit-community.github.io/qiskit-optimization/tutorials/08_cvar_optimization.html.
48. Kungurtsev V, Korpas G, Marecek J, Zhu EY. Iteration complexity of variational quantum algorithms. Quantum. 2024;8:1495. doi:10.22331/q-2024-10-10-1495.
49. Amaro D, Modica C, Rosenkranz M, Fiorentini M, Benedetti M, Lubasch M. Filtering variational quantum algorithms for combinatorial optimization. Quantum Sci Technol. 2022;7(1):15021. doi:10.1088/2058-9565/ac3e54.
50. Cerezo M, Sone A, Volkoff T, Cincio L, Coles PJ. Cost function dependent barren plateaus in shallow parametrized quantum circuits. Nat Commun. 2021;12(1):1791. doi:10.1038/s41467-021-21728-w.
51. Mastropietro D, Korpas G, Kungurtsev V, Marecek J. Parallel variational quantum algorithms with gradient-informed restart to speed up optimisation in the presence of barren plateaus. arXiv:2311.18090. 2024. doi:10.48550/arXiv.2311.18090.