Otimização Quântica Híbrida (QAOA) para Problema Simples Inspirado no Posicionamento de Turbinas Eólicas: Uma Prova de Conceito

Hybrid Quantum Optimization (QAOA) for a Simple Problem Inspired by Wind Turbine Placement: A Proof of Concept

Marcos Vinícius Cândido Henriques[1]

[1]Universidade Federal Rural do Semi-Árido / Campus Angicos-RN

VIII ⟨WECIQ | WCQ⟩
8-12 DEC/25

Summary

Quantum at the Dunes

Classical simulation: resources vs qubits (AWS)

How fast do resources explode to classically simulate circuits with dozens of qubits?

Number of qubits	Required memory (GiB)	Number of instances	Total number of cores
36	2.199	16	576
37	4.398	32	1.152
38	8.796	64	2.304
39	17.592	128	4.608
40	35.184	256	9.216
41	70.369	512	18.432
42	140.737	1.024	36.864
43	281.475	2.048	73.728
44	562.950	4.096	147.456

Simulating 44-Qubit quantum circuits using AWS ParallelCluster (Dr. Fabio Baruffa; Pavel Lougovski), 30 AUG 2022, AWS Center for Quantum Computing, AWS.

Context and motivation

- Growing share of **wind energy** in RN and in Brazil.
- **Wake effect**: downstream power reduction and turbulence increase.
- Turbine siting decisions directly impact **capacity factor** and **LCOE**.
  - **Capacity factor**: energy generated / energy at 100% rated power.
  - **LCOE**: levelized cost per MWh over project lifetime.

Turbinas da Vattenfall no Mar do Norte (2011). Crédito: NOAA.

Offshore layout example

Journal of Ocean Engineering and Marine Energy (2024) 10:773–782
https://doi.org/10.1007/s40722-024-00345-x

RESEARCH ARTICLE

Check for updates

Possible application of quantum computing in the field of ocean engineering: optimization of an offshore wind farm layout with the Ising model

Hiroshi Kagemoto[1]

Received: 15 May 2024 / Accepted: 12 August 2024 / Published online: 31 August 2024

Abstract
As a possible application of quantum computing in the field of ocean engineering, an attempt is made to identify the optimum layout of an offshore wind farm with the quasi-quantum annealing method while assuming a wake model that conforms to the standard Jensen–Katic wake model but can be implemented in the Ising model. The optimum layouts obtained in the present study are compared with those identified with other conventional methods such as the genetic algorithm. It is confirmed that the quasi-quantum annealing method works as well as other methods in identifying the optimum layouts, while the optimum layout of an offshore wind farm composed of even more than 100 wind turbines can be identified with reasonable CPU time.

Keywords Quantum computing · Simulated annealing · Quasi-annealing method · Wind farm layout · Ising model

Source: H. Kagemoto (2024) – offshore wind farm layout optimized via Ising model.

Wake model

- **Standard (Jensen–Katic)**: velocity deficit decays with downstream distance and wake expansion. Typical form:

$$1 - \frac{u}{u_0} = \frac{2a}{\left(1 + k\,x/r_0\right)^2}$$

  where u_0: upstream speed (inflow), u: speed at downstream x, a: axial induction factor, k: wake decay coefficient, r_0: rotor radius, x: downstream distance.

- **In this work (proof of concept)**: we adopt a **simple model** in which available power **decays linearly** with distance from the upstream turbine (up to a fixed range). When wakes overlap, penalties are added.

- **Goal**: have a **didactic scenario** to demonstrate an application of QAOA, with penalties easy to tune and fast execution.

What is QAOA? (1/2)

- QAOA = *Quantum Approximate Optimization Algorithm*.

QAOA is inspired by the Adiabatic Theorem.

Adiabatic theorem (Born & Fock, 1928)

A physical system remains in its instantaneous eigenvector if a perturbation acts on it **slowly** enough and if there is an **energy gap** between the corresponding eigenvalue and the rest of the Hamiltonian spectrum.[1]

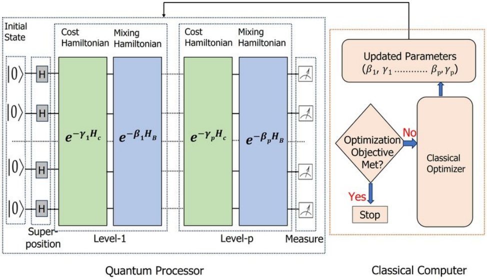
Direct consequence of the adiabatic theorem

If we start in the **ground state** of a time-dependent Hamiltonian and the Hamiltonian evolves **slowly** enough, given sufficient time, the final state will be the **ground state** of the final Hamiltonian.

[1] Born, M.; Fock, V. A. (1928). "Beweis des Adiabatensatzes". Zeitschrift für Physik A. 51 (3–4): 165–180.

What is QAOA? (2/2)

- Variational algorithm for **combinatorial optimization** on noisy quantum hardware (NISQ).
- Parameterized state: alternates layers of **cost** (H_C) and **mixer** (H_M) with depth p.
- **Hybrid**: the parameters (γ, β) are **optimized on a classical computer**; the circuit is evaluated on a quantum device/simulator.
- Measurements return candidate *bitstrings*; we pick the one(s) with best cost value.



Wake model and conflict graph

- Conflict graph $G = (V, \mathcal{W})$.
  - $V = \{0, \dots, n-1\}$: grid positions.
  - $\mathcal{W} = \{(i,j) \mid i < j, w_{ij} > 0\}$: pairs under wake interaction.

$$B(x) = \sum_{i \in V} s_i x_i \quad \text{(benefit)}$$

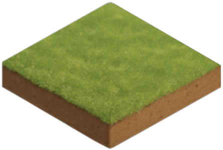
$$P(x) = \sum_{(i,j) \in \mathcal{W}} w_{ij}\, x_i x_j \quad \text{(penalties)}$$

QUBO (Quadratic Unconstrained Binary Optimization) cost to minimize:

$$C(x) = -B(x) + P(x)$$

Qubits: states representing absence or presence of a turbine

$$|0\rangle$$



$$|1\rangle$$

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

1-qubit term $H_i^{(1)}$ (reward)

Mapping from binary variable:

$$x_i = \frac{1 - Z_i}{2}$$

Term definition for position i:

$$\boxed{H_i^{(1)} = -\frac{s_i}{2}\left(I - Z_i\right)}$$

- I: identity matrix.
- Z_i: Pauli-Z operator on qubit i.
- s_i: benefit/*score* associated with position i.
- Action of Z in the computational basis:
  - $Z|0\rangle = |0\rangle$ (eigenvalue $+1$)
  - $Z|1\rangle = -|1\rangle$ (eigenvalue -1)

Energy contributions of term $H_i^{(1)}$:

- $E(|0\rangle) = 0$
- $E(|1\rangle) = -s_i$
  - state $|1\rangle$ is rewarded

2-qubit term (wake penalty in $|11\rangle$)

Term definition for pair (i, j):

$$\boxed{\begin{aligned} H_{ij}^{(2)} &= \frac{w_{ij}}{4}\left(Z_i Z_j - Z_i - Z_j + I\right) \\ &\equiv \frac{w_{ij}}{4}(I - Z_i)(I - Z_j) \end{aligned}}$$

- I: identity matrix.
- Z_i, Z_j: Pauli-Z operators on qubits i and j.
- $Z_i Z_j$: tensor product $Z \otimes Z$ acting on the pair (i, j).
- w_{ij}: wake penalty for pair $(i, j) \in \mathcal{W}$.

Energy contributions of term $H_{ij}^{(2)}$ per state of pair (i, j):

- $E(|00\rangle) = E(|01\rangle) = E(|10\rangle) = 0$
- $E(|11\rangle) = w_{ij}$
  - penalizes only the state $|11\rangle$.

Total cost Hamiltonian

Sum of the 1-qubit and 2-qubit terms built in the code:

$$H_C = \sum_{i \in V} \frac{s_i}{2} (Z_i - I) + \sum_{(i,j) \in \mathcal{W}} \frac{w_{ij}}{4} \Big(Z_i Z_j - Z_i - Z_j + I \Big)$$

- I: identity.
- Z_i, Z_j: Pauli-Z operators on qubits i and j.
- $Z_i Z_j = Z \otimes Z$ is the tensor product on pair (i, j).
- s_i: benefit/*score* of the turbine at position i.
- w_{ij}: wake penalty for pair $(i, j) \in \mathcal{W}$.
- V: set of positions (vertices); $\mathcal{W}$: edges with wake.

Code (Python + Qiskit): cost Hamiltonian

Python example using Qiskit.

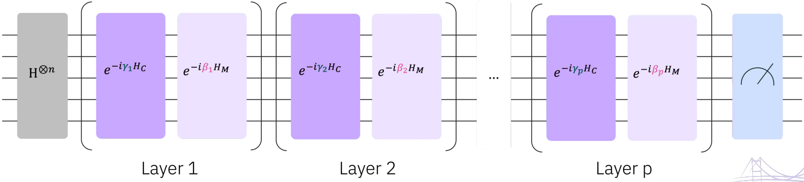
```python
def create_cost_hamiltonian():
  pauli_list = []
  const_offset = 0.0

  for i in range(optimizer.n_positions):
    pauli_list.append(("Z", [i], score[i]/2))
    const_offset += -score[i]/2

  for (i, j), wake_penalty in wake_penalties.items():
    pauli_list.append(("ZZ", [i, j],  wake_penalty/4))
    pauli_list.append(("Z",  [i],    -wake_penalty/4))
    pauli_list.append(("Z",  [j],    -wake_penalty/4))
    const_offset += wake_penalty/4

  if abs(const_offset) > 0:
    pauli_list.append(("I", [], const_offset))

  return SparsePauliOp.from_sparse_list
    (pauli_list, num_qubits=optimizer.n_positions)
```

Hamiltonian ⇒ quantum circuit

- The Hamiltonian H_C encodes the quantum definition of the problem; from it we build a **circuit** to sample good solutions.
- QAOA applies **alternating layers** of **cost** and **mixer** operators in the circuit.
- Start in the state $H^{\otimes n}|0\rangle$ and drive the system to the ground state of H_C by applying $e^{-i\,\gamma_k H_C}$ and $e^{-i\,\beta_k H_M}$, with $\gamma_1, \ldots, \gamma_p$ and $\beta_1, \ldots, \beta_p$.
- H_M: **mixer** Hamiltonian; creates exploration between configurations.

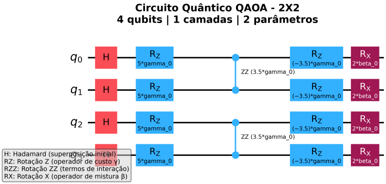
$$H_M = \sum_{i \in V} X_i$$

- The circuit is **parameterized** by $\{\gamma_i, \beta_i\}$; we test different values and sample the resulting state.

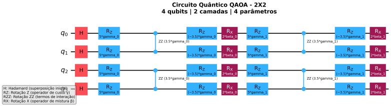


Camadas alternadas de custo e mixer (QAOA)

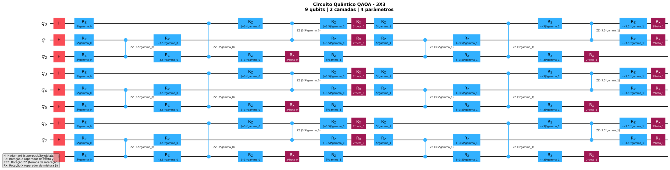
Quantum circuits (QAOA)



2x2, 4 qubits, 1 layer



2x2, 4 qubits, 2 layers



3x3, 9 qubits, 2 layers

Qiskit Primitives strategy in QAOA

Primitive roles

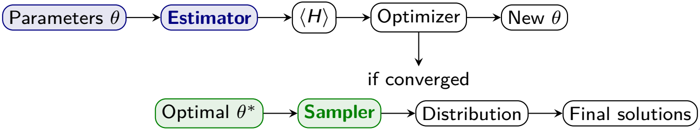
- **Estimator**: computes expectation values of observables $\Rightarrow$ returns **real numbers**.
- **Sampler**: samples quantum states $\Rightarrow$ returns **bitstring distributions**.

During optimization loop (cost function)

- **Estimator only**
- Computes $\langle\psi|H_{\text{cost}}|\psi\rangle$
- Classical optimizer receives scalar value

After convergence (results analysis)

- **Sampler only** with optimal parameters
- **Shots**: samples per run (e.g., 1024, 2048, 4096, 8192)
- Probabilities = counts/shots; more shots $\rightarrow$ lower variance
- Inspect most likely bitstrings and classical cost



Scenarios and parameters

- **Grids evaluated**: 2×3, 3×3, 4×4.
- **Classical optimizer**: COBYLA (up to 200 iterations or convergence).
- **Parameters**: depth p, COBYLA initial step size/trust-region radius (`rhobeg`), and γ, β ranges.
- **Simulations**: combinations of $p \in \{1, 2, 3\}$ and `rhobeg` $\in \{0.3, 0.5, 0.7\}$.
- **Runs per setting**: 3 repetitions per $(p, \texttt{rhobeg})$.
- **Shots**: 1000 samples per circuit evaluation.
- **Reward** (`score`) per installed turbine: **5** (2×3, 3×3 and 4×4).
- **Wake penalty** (scaled by grid size):
  - Grid 2×3: `base_penalty`=3.0, `distance_decay`=1.0
  - Grid 3×3: `base_penalty`=8.0, `distance_decay`=1.0
  - Grid 4×4: `base_penalty`=12.0, `distance_decay`=4.0

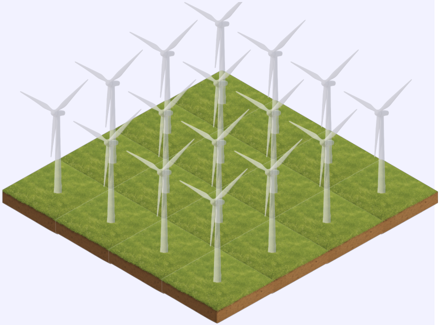
Initial configuration

- All possible solutions in a uniform **superposition**.
- Each grid position is mapped to one **qubit**.
- n **qubits** can represent 2^n states.
  - 3x3 ($n = 9$): $2^9 = 512$ states
  - 4x4 ($n = 16$): $2^{16} = 65{,}536$ states
- Qubit for each position i:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

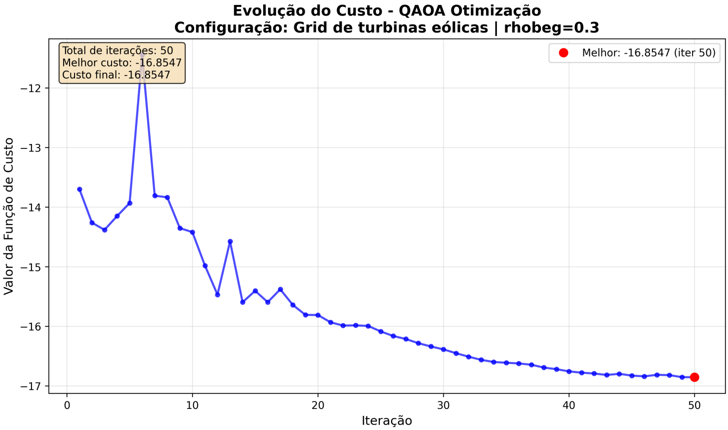
Initial state of the system:

$$|\psi_0\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$



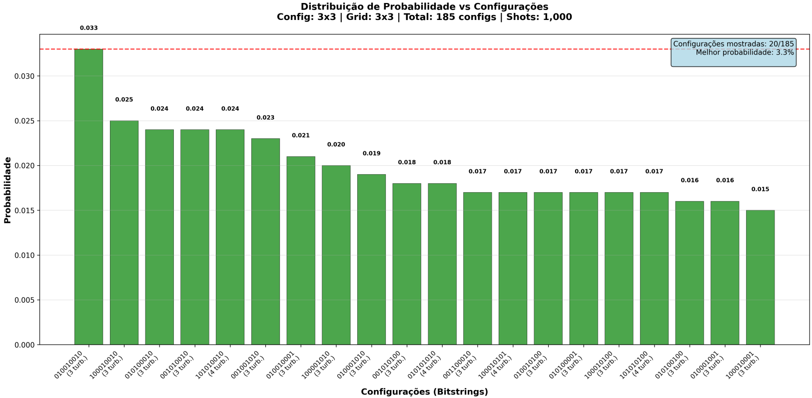
Initial superposition state (4x4).

Cost evolution across iterations

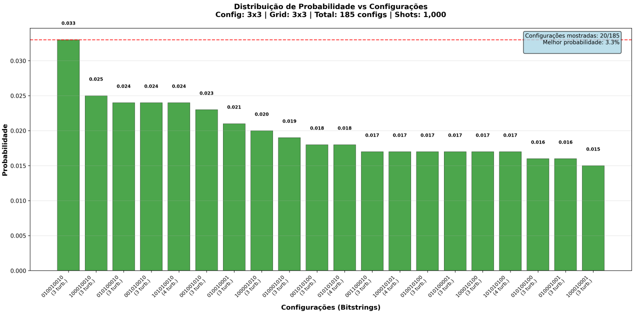


Example on the **3x3 grid**: cost trajectory along optimizer iterations.

Probability distribution (3x3)

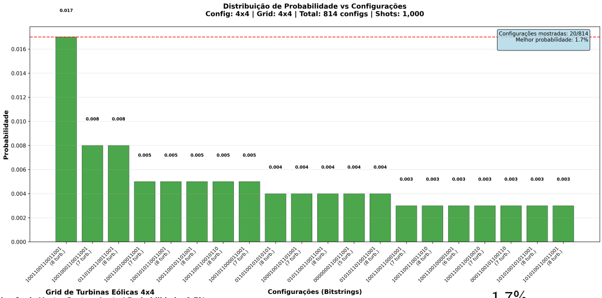


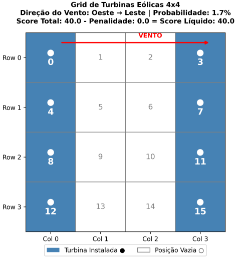
Probability distribution (3x3)



3,3%	2,5%	2,4%	2,4%
			
$\|010010010\rangle$	$\|100010010\rangle$	$\|010100010\rangle$	$\|001010010\rangle$

Probability distribution (4x4) and layouts





Optimized layout (4x4) — run

1,7%

$|1001100110011001\rangle$

Optimized layout (4x4) and costs

$|1001100110011001\rangle$

QAOA results — CPU simulation (Grid 2×3, 6 qubits)

Layers	rhobeg (COBYLA)	Mean Prob. (%)*	Max Prob. (%)	Score Mean	Score Max	Time (s)
1	0.3	32.7	71.0	18.3	20.0	0.62
1	0.5	15.7	21.0	18.0	18.0	0.62
1	0.7	11.0	12.0	12.0	17.0	0.61
2	0.3	68.3	70.0	20.0	20.0	0.72
2	0.5	39.0	96.0	19.7	20.0	0.82
2	0.7	39.7	67.0	19.3	20.0	0.74
3	0.3	56.3	96.0	19.7	20.0	0.87
3	0.5	86.7	93.0	20.0	20.0	0.85
3	0.7	52.3	65.0	20.0	20.0	0.81

Analysis by **layers** and `rhobeg` (COBYLA).

* Mean probability of the best bitstring.

QAOA results — CPU simulation (Grid 3×3, 9 qubits)

Layers	rhobeg (COBYLA)	Mean Prob. (%)*	Max Prob. (%)	Score Mean	Score Max	Time (s)
1	0.3	4.3	5.0	10.0	11.0	0.70
1	0.5	4.3	5.0	10.0	11.0	0.71
1	0.7	4.3	5.0	11.7	15.0	0.67
2	0.3	13.3	18.0	13.0	15.0	0.85
2	0.5	10.0	18.0	13.0	15.0	0.85
2	0.7	4.3	5.0	12.7	15.0	0.85
3	0.3	12.0	13.0	13.0	15.0	0.95
3	0.5	13.0	14.0	13.0	15.0	0.95
3	0.7	5.0	7.0	13.7	15.0	0.97

Analysis by **layers** and `rhobeg` (COBYLA).

* Mean probability of the best bitstring.

QAOA results — CPU simulation (Grid 4 × 4, 16 qubits)

Layers	rhobeg (COBYLA)	Mean Prob. (%)*	Max Prob. (%)	Score Mean	Score Max	Time (s)
1	0.3	2.0	2.0	20.7	30.0	3.28
1	0.5	1.7	2.0	16.0	20.0	1.74
1	0.7	1.7	2.0	16.7	17.0	1.48
2	0.3	2.0	2.0	30.0	36.0	7.35
2	0.5	1.3	2.0	6.3	13.0	14.29
2	0.7	2.7	4.0	21.3	31.0	4.40
3	0.3	3.7	7.0	35.3	40.0	14.82
3	0.5	32.3	92.0	34.7	40.0	12.88
3	0.7	2.3	3.0	26.3	40.0	13.35

Results by **layers** and `rhobeg` (COBYLA). Timings on Intel Core i3-10100.

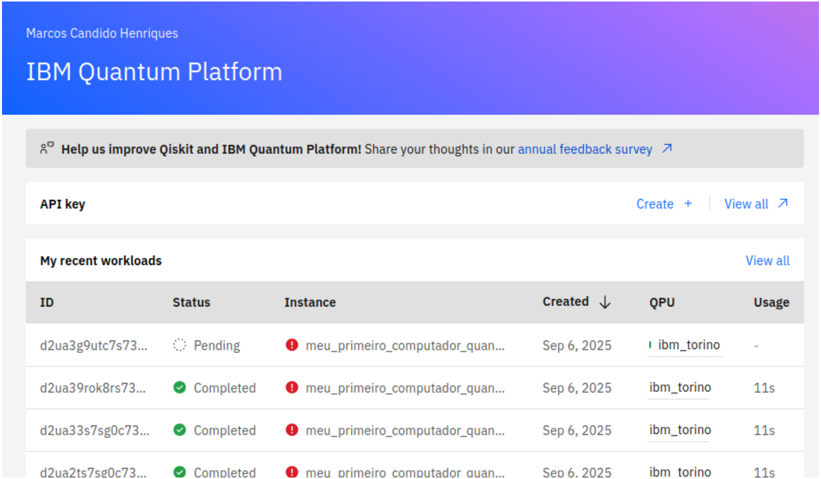
* Mean probability of the best bitstring.

Performance comparison

Grid	Qubits	Score Max	Optimal Layers	Optimal rhobeg	Prob. Max (%)	Time Mean (s)
2x3	6	20.0	1	0.3	71.0	0.62
3x3	9	15.0	1	0.7	5.0	0.67
4x4	16	40.0	3	0.3	7.0	14.82

Best results per grid according to Table IV of the paper.

Execution on IBM Quantum Platform


Marcos Candido Henriques

IBM Quantum Platform

Help us improve Qiskit and IBM Quantum Platform! Share your thoughts in our annual feedback survey

API key Create + | View all

My recent workloads View all

ID	Status	Instance	Created	QPU	Usage
d2ua3g9utc7s73...	Pending	meu_primeiro_computador_quan...	Sep 6, 2025	ibm_torino	-
d2ua39rok8rs73...	Completed	meu_primeiro_computador_quan...	Sep 6, 2025	ibm_torino	11s
d2ua33s7sg0c73...	Completed	meu_primeiro_computador_quan...	Sep 6, 2025	ibm_torino	11s
d2ua2ts7sg0c73...	Completed	meu_primeiro_computador_quan...	Sep 6, 2025	ibm_torino	11s

Challenges in 5x5 grids

- **IBM hardware topology**
  - Each qubit connects directly to at most 3 neighbors.
  - Extra links require **SWAP** operations, increasing quantum errors.
- **Classical simulation limits**
  - Systems with 25 qubits already demand high computational power.
  - Memory and processing cost grow **exponentially**.
- **Circuit complexity**
  - Greater circuit depth means more quantum gates applied.
  - Accumulated gate errors and **decoherence** reduce result fidelity.

Conclusions

- Built a **QAOA layout model** for **wind turbines** (2x3 to 4x4 grids) in **Qiskit** with **directional wake penalties** that fade with distance.
- On small grids, QAOA finds good layouts that trade **production** vs **wake**; more layers help but cost more time.
- **Runtime** rises fast as the grid grows.

Possible future work

- Scale to **larger grids**.
- Refine the **wake model** and penalty calibration.
- Define a **heterogeneous production score** per site.
- Test **optimizers** beyond ***COBYLA*** (e.g., ***SPSA***, *Nelder–Mead*, *L-BFGS-B*, *SLSQP*, *ADAM*).
- Investigate how parameter **ranges** (γ, β) affect the search for solutions.
- Search for suitable **layer counts** for each problem configuration.
- Post-processing to **clean up** imperfect solutions.
- Study **noise** and run on **quantum hardware**.
- Test **mixer Hamiltonians** more complex than $H_M = \sum_i X_i$.
- Apply ***warm-start*** and ***layerwise*** (layer-by-layer optimization) techniques.

References

- R. J. Barthelmie et al., “Modelling and measuring flow and wind turbine wakes in large wind farms offshore,” Wind Energy, 12(5):431–444, 2009.
- H. Kagemoto, “Possible application of quantum computing in the field of ocean engineering: optimization of an offshore wind farm layout with the Ising model,” Journal of Ocean Engineering and Marine Energy, 10:773–782, 2024.
- G. Kochenberger et al., “The unconstrained binary quadratic programming problem: a survey,” Journal of Combinatorial Optimization, 28(1):58–81, 2014.
- F. Glover, G. Kochenberger, Y. Du, “Quantum bridge analytics I: a tutorial on formulating and using QUBO models,” 4OR, 17(4):335–371, 2019.
- E. Farhi, J. Goldstone, S. Gutmann, “A *Quantum Approximate Optimization Algorithm*,” arXiv:1411.4028, 2014.
- Qiskit Community, “Qiskit: An Open-Source Framework for Quantum Computing,” 2017. DOI: 10.5281/zenodo.2562110. Disponível em: `https://github.com/Qiskit/qiskit` (acesso em agosto de 2025).
- J. Preskill, “Quantum computing in the NISQ era and beyond,” Quantum, 2:79, 2018.

Acknowledgments, contact, and repository

Acknowledgments

- Institutional support (UFERSA).
- Qiskit community and project contributors.
- Organizers of VIII WECIQ/WCQ 2025.

Contact and repository

- Repository: `https://github.com/vinirn/qaoa_wind`

- Contact: `viniciuscandido@ufersa.edu.br`

Thank you!