Encontro Potiguar de FÍSICA

11 e 12/09/2025 - Hotel Serrano - Martins/RN

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

Marcos Vinícius Cândido Henriques[1]

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

Martins-RN, 2025

Sumário

Motivação e objetivos

Fundamentos

Metodologia

Resultados

Conclusões e próximos passos

Demonstração

Agradecimentos e referências

Contexto e motivação

- Crescente participação da energia eólica no RN e no Brasil.
- Efeito de esteira (wake): redução da potência e aumento da turbulência a jusante.
- Decisão de posicionamento das turbinas impacta diretamente **fator de capacidade** e **LCOE**.

Objetivos do trabalho

- Formular o problema de layout como otimização combinatória.
- Aplicar o **QAOA** para maximizar geração com penalidades por interferência.
- Avaliar cenários simples (2x3, 3x3, 4x4) e discutir escalabilidade.

Código: Hamiltoniano de custo

```
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)
```

QAOA em linhas gerais

- Alternância entre operadores de **custo** e **mistura** com profundidade p.
- Parâmetros (γ, β) otimizados por rotina clássica (loop híbrido).
- Medidas fornecem bitstrings candidatos a layouts viáveis.

Modelo de esteiras e grafo de conflitos

- Penalidade por pares de turbinas com **interferência** acima de limiar (matriz de interferência).
- Mapeamento para um grafo: vértices = posições; arestas = penalidades/esteiras.
- Função custo: termo de **benefício** por turbina ativa e **penalidade** por conflitos.

Hamiltonianos e implementação (1/2)

Variáveis e custo clássico:

$x_i \in \{0,1\}$ $(i \in V)$	Variável binária indicando turbina ativa na posição i.
$B(x) = \sum_{i \in V} b_i x_i$	Benefícios (ganhos) por turbinas ativas; b_i é o ganho da posição i.
$P(x) = \sum_{(i,j) \in \mathcal{W}} \lambda_{ij}\, x_i x_j$	Penalidades por pares com esteira; λ_{ij} é a penalidade para $(i,j) \in \mathcal{W}$.
$C(x) = -B(x) + P(x)$	Custo total a minimizar.

Seja $G = (V, \mathcal{W})$ o grafo de conflitos, com $V = \{0, \ldots, n-1\}$ e

$$\mathcal{W} = \{\, (i,j) \mid i < j,\ w_{ij} > 0 \,\}, \qquad w_{ij} = \mathrm{wake\_penalty}(i,j) \geq 0.$$

Hamiltonianos e implementação (2/2)

Hamiltoniano de custo:

$$H_C = -\sum_{i \in V} b_i \, \frac{1-Z_i}{2} + \sum_{(i,j) \in \mathcal{W}} \lambda_{ij} \, \frac{(1-Z_i)(1-Z_j)}{4}$$

Forma equivalente (Ising):

$$H_C = \text{const} + \sum_{i \in V} h_i Z_i + \sum_{(i,j) \in \mathcal{W}} J_{ij} Z_i Z_j, \quad J_{ij} = \frac{\lambda_{ij}}{4}$$

Mapeamento para operadores de Pauli:

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

Mixer e estado inicial:

$$H_M = \sum_{i \in V} X_i, \qquad |\psi_0\rangle = |+\rangle^{\otimes n}$$

Termo de 1-qubit (recompensa)

Notação: $s_i = \texttt{score}[i]$.
Definição do termo por posição i:

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

- I: matriz identidade; Z_i: operador de Pauli-Z no qubit i.
- s_i: benefício/*score* associado à posição i.
- Efeitos: $E(|0\rangle) = 0$, $E(|1\rangle) = -s_i \Rightarrow |1\rangle$ é recompensado.

Termo de 2-qubits (esteira em $|11\rangle$)

Definição do termo para o par (i, j):

$$H_{ij}^{(2)} = \frac{w_{ij}}{4}\left(Z_i Z_j - Z_i - Z_j + I\right)$$
$$\equiv \frac{w_{ij}}{4}(1 - Z_i)(1 - Z_j)$$

- I: matriz identidade; Z_i, Z_j: operadores de Pauli-Z nos qubits i e j.
- $Z_i Z_j$: produto tensorial $Z \otimes Z$ agindo no par (i, j).
- w_{ij}: penalidade de esteira para o par $(i, j) \in \mathcal{W}$.

Contribuições por estado do par (i, j): penaliza apenas $|11\rangle$ com custo w_{ij}; demais estados têm custo zero.

Hamiltoniano total (Pauli-Z)

Soma dos termos de 1-qubit e 2-qubits construídos no código:

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

- I: identidade; Z_i e $Z_i Z_j$: Pauli-Z e seu produto no(s) qubit(s) indicados.
- s_i: benefício/*score* da turbina na posição i; w_{ij}: penalidade de esteira do par $(i, j) \in \mathcal{W}$.
- V: conjunto de posições (vértices); $\mathcal{W}$: arestas com wake, conforme definido.

Observação (forma binária equivalente, com $x_i = (1 - Z_i)/2$):

$$H \equiv -\sum_{i \in V} s_i \, x_i + \sum_{(i,j) \in \mathcal{W}} w_{ij} \, x_i x_j + \text{const}$$

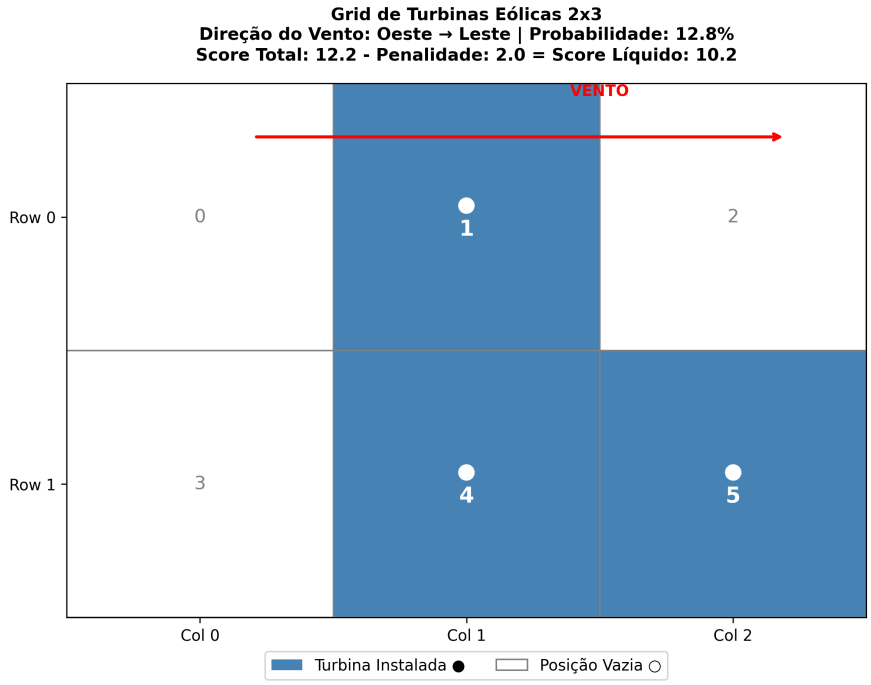
Pipeline do experimento

- Carrega `config*.json` com grid, direção do vento e pesos.
- Constrói operadores, inicializa QAOA e define otimizador clássico.
- Executa avaliação via EstimatorV2 e salva gráficos em `images/`.

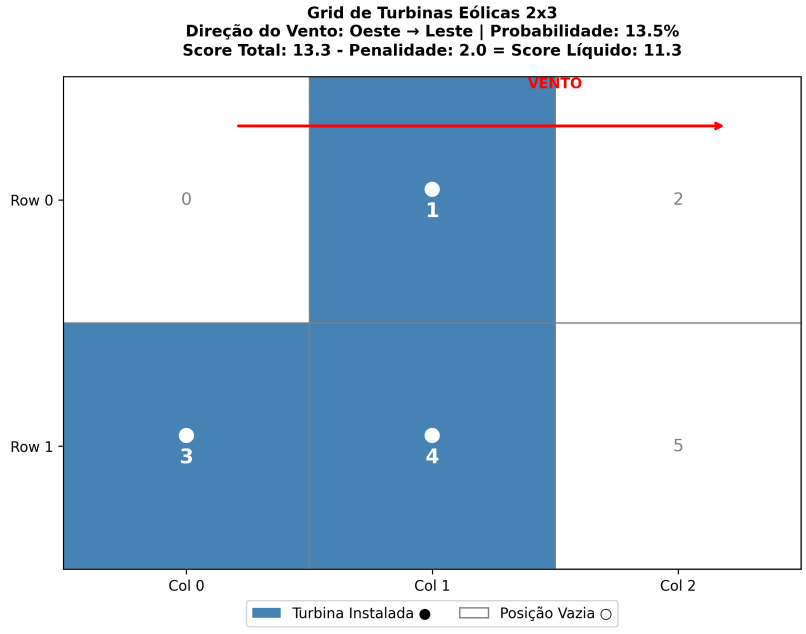
Cenários e parâmetros

- Grades avaliadas: 2x3, 3x3, 4x4; sementes fixas para reprodutibilidade.
- Profundidade p, *step size* (`rhobeg`) e número de avaliações controlam o custo.
- Métricas: energia total aproximada, número de conflitos e custo final.

Layouts (2x3) e exemplo

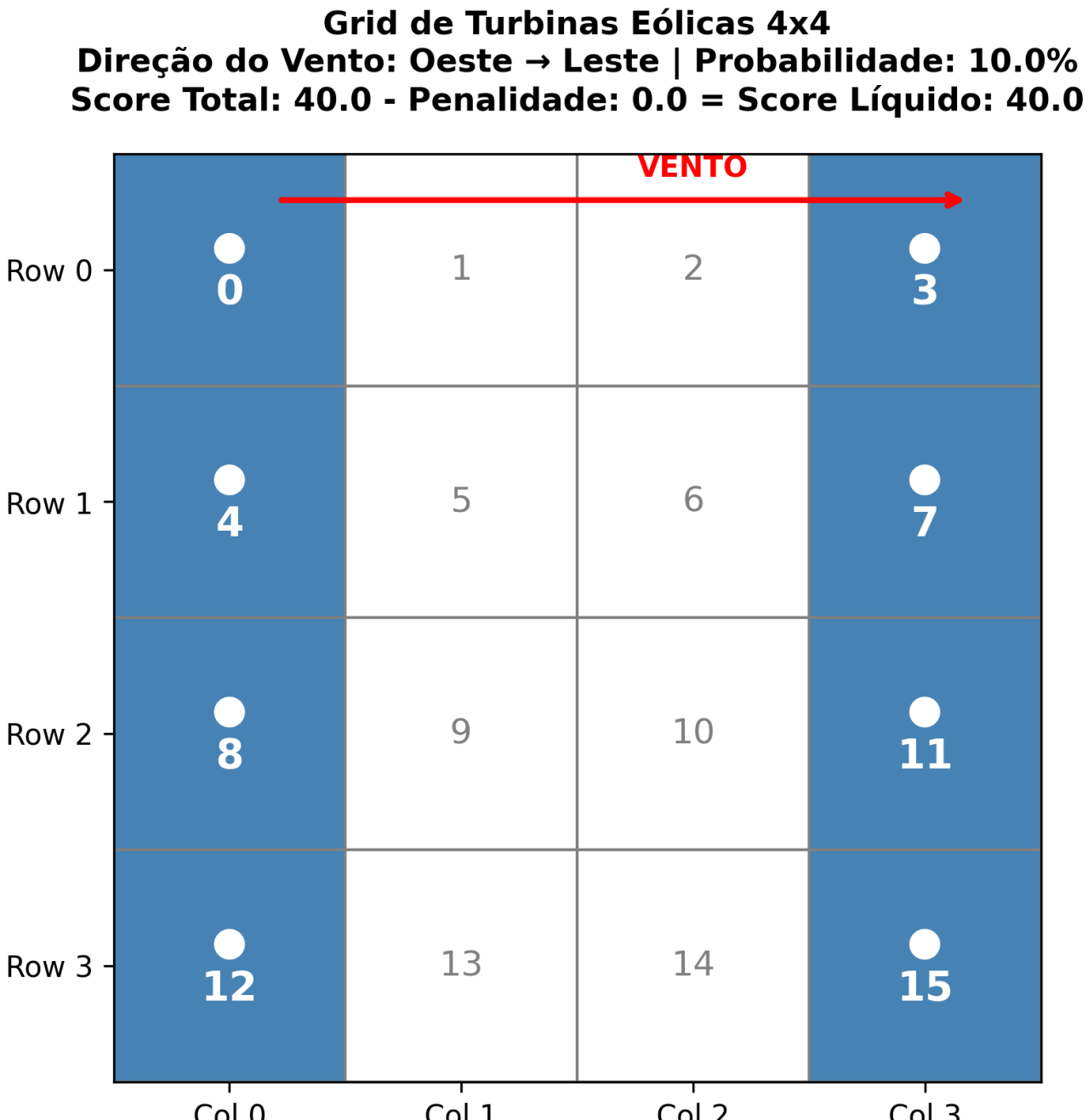


Visualização do layout (2x3) — execução A



Visualização do layout (2x3) — execução B

Layout otimizado (4x4) e custos



Discussão

- Soluções plausíveis em grades pequenas; sensíveis à escolha de pesos e limiares.
- **Trade-off** entre qualidade e tempo de execução conforme p e avaliações.
- Escalabilidade: crescimento do espaço de busca exige heurísticas/estruturas adicionais.

Conclusões

- QAOA viabiliza formulação híbrida para o layout de turbinas com esteiras simplificadas.
- Pipeline reproduzível com `config*.json` e salvamento automático de resultados.

Trabalhos futuros

- Refinar modelo de esteiras e calibração de penalidades.
- Estudo de ruído e execução em *backends* reais.
- Técnicas de *warm-start*, *layerwise* e ajustes de otimizadores.

Como executar

```
# Ambiente e dependencias
python3 -m venv qiskit_env && source qiskit_env/bin/acti
pip install -r requirements.txt

# Execucao com configuracao padrao
./run_qaoa.sh

# Outros cenarios
./run_qaoa.sh config_3x3.json
python qaoa_turbinas.py -c config_4x4.json
python qaoa_turbinas.py --list-configs
```

Agradecimentos

- Apoio institucional (UFERSA).
- Comunidade Qiskit e contribuidores do projeto.

Referências

- Farhi, Goldstone, Gutmann. A Quantum Approximate Optimization Algorithm (2014).
- Documentação Qiskit e tutoriais de variational algorithms.
- Revisões sobre modelos de esteiras (ex.: Jensen/Park) e layout eólico.

Contato e repositório

- Repositório: `https://github.com/`
- Contato: [e-mail/QR code]