

# Passo a Passo para Aplicar QAOA ao Problema da Mochila

## 1 Introdução

O problema da mochila (*Knapsack Problem*) é um problema de otimização combinatória onde, dado um conjunto de  $n$  itens, cada um com peso  $w_i$  e valor  $v_i$ , e uma mochila com capacidade máxima  $W$ , o objetivo é selecionar um subconjunto de itens que maximize o valor total  $\sum v_i x_i$  sujeito à restrição  $\sum w_i x_i \leq W$ , onde  $x_i \in \{0, 1\}$  indica se o item  $i$  é selecionado. Este documento descreve os passos para resolver o problema da mochila usando o **QAOA puro** (*Quantum Approximate Optimization Algorithm*) e o **LR-QAOA** (*Low-Rank QAOA*), uma variante que utiliza uma inicialização quente (*warm-start*) baseada em uma solução clássica. Os passos são apresentados de forma didática para facilitar a implementação.

## 2 Passos para QAOA Puro

### 1. Formular o problema como QUBO ou Ising

- Converta o problema da mochila em uma função objetivo quadrática sem restrições explícitas.
- Função objetivo: Maximizar  $\sum_{i=1}^n v_i x_i - \lambda (\sum_{i=1}^n w_i x_i - W)^2$ , onde  $\lambda$  é um parâmetro de penalidade (ex.:  $\lambda = 100$ ).
- Expanda o termo de penalidade para obter termos quadráticos:  $\sum_{i=1}^n \sum_{j=1}^n \lambda w_i w_j x_i x_j$  mais termos lineares.
- Resultado: Um modelo QUBO com variáveis binárias  $x_i \in \{0, 1\}$ .

### 2. Construir o Hamiltoniano de Custo ( $H_C$ )

- Mapeie o QUBO para operadores Pauli, onde  $x_i = \frac{1-Z_i}{2}$  (modelo Ising).
- Para o termo de valor  $v_i x_i$ : Use  $\frac{v_i}{2}(I - Z_i)$ .
- Para a penalidade: Inclua termos como  $\lambda w_i w_j \frac{(I-Z_i)(I-Z_j)}{4}$ , expandidos em operadores Pauli-Z.
- Represente  $H_C = \sum$  coeficientes  $\cdot$  produtos de  $Z_i Z_j$ .
- Use bibliotecas como Qiskit (`SparsePauliOp`) para implementar.

### 3. Construir o Hamiltoniano de Mistura ( $H_M$ )

- No QAOA padrão,  $H_M = \sum_{i=1}^n X_i$  (operadores Pauli-X).
- Para a mochila, considere um mixer personalizado (*Quantum Weighted Mixer*):  $H_M = \sum_{i=1}^n (c_i X_i + d_i Y_i)$ , onde  $c_i, d_i$  são ponderados pela razão  $v_i/w_i$ , normalizada pelo maior valor.
- Exemplo:  $c_i = \frac{v_i/w_i}{\max(v_i/w_i)}$ ,  $d_i = 0.5 \cdot c_i$ .

### 4. Definir os parâmetros do algoritmo

- Escolha  $p$ , o número de camadas (ex.:  $p = 5$ ).
- Defina vetores de ângulos:  $\gamma = (\gamma_1, \dots, \gamma_p)$  para  $H_C$ , e  $\beta = (\beta_1, \dots, \beta_p)$  para  $H_M$ .
- Inicialize  $\gamma_k, \beta_k$  heurísticamente (ex.:  $\gamma_k \propto k/p \cdot \pi$ ,  $\beta_k \propto (1 - k/p) \cdot \pi$ ) ou otimize classicamente.

### 5. Criar o circuito quântico

- Inicialize com superposição uniforme: Aplique Hadamard ( $H$ ) em cada um dos  $n$  qubits, criando o estado  $|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$ .
- Para cada camada  $k = 1, \dots, p$ :
  - Aplique a evolução  $e^{-i\gamma_k H_C}$  (porta de fase baseada em  $H_C$ ).
  - Aplique a evolução  $e^{-i\beta_k H_M}$  (porta de mistura baseada em  $H_M$ ).
- Use aproximações como Lie-Trotter para implementar as evoluções.
- Adicione medições em todos os qubits.

## 6. Executar o circuito

- Use um simulador quântico (ex.: Qiskit BasicSimulato) ou hardware real.
- Execute com muitos shots (ex.: 7000) para obter uma distribuição de bitstrings.
- Otimize  $\gamma, \beta$  classicamente (ex.: usando gradiente ou COBYLA) para maximizar  $\langle H_C \rangle$ .

## 7. Pós-processamento e avaliação

- Para cada bitstring medida:
  - Calcule peso ( $\sum w_i x_i$ ) e valor ( $\sum v_i x_i$ ).
  - Verifique se é válida ( $\sum w_i x_i \leq W$ ).
- Selecione a melhor solução válida (maior valor).
- Compare com a solução ótima (se disponível, ex.: via programação dinâmica) para calcular a taxa de aproximação ( $\text{valor}_{\text{QAOA}}/\text{valor}_{\text{ótimo}}$ ).