Efficient Approaches to the Bin Packing Problem

Optimization with Integer Linear Programming and Brute Force

Haris Ansari

Indian Institute of Technology Jodhpur

November 18, 2024

Introduction

Problem Statement:

- Given n items with weights $w_1, w_2, \ldots, w_n$, and bins with capacity C.
- Objective: Minimize the number of bins used while ensuring:
  - Each item is assigned to exactly one bin.
  - Total weight in each bin does not exceed C.

ILP Formulation

Decision Variables:

- x_{ij}: 1 if item i is placed in bin j, 0 otherwise.
- y_j: 1 if bin j is used, 0 otherwise.

Objective Function:

$$\text{Minimize} \sum_{j=1}^{m} y_j$$

Constraints:

1. Each item is assigned to one bin:

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i$$

2. Total weight in a bin does not exceed capacity:

$$\sum_{i=1}^{n} w_i x_{ij} \leq C y_j \quad \forall j$$

Bin Packing Problem Formulation

Problem:

- **Weights:** `weights` $= [3, 2, 3, 4]$, **Bin Capacity:** $C = 6$ (max weight $+ 2$).
- **Objective:** Minimize the number of bins used.

Variables:

- $x_{ij} \in \{0, 1\}$ (Item i in Bin j), $y_j \in \{0, 1\}$ (Bin j used).

Objective Function:

$$\text{Minimize: } y_0 + y_1$$

Constraints:

- Item Assignment: $x_{i0} + x_{i1} = 1$ for each item i.
- Bin Capacity: $3x_{0j} + 2x_{1j} + 3x_{2j} + 4x_{3j} - 6y_j \leq 0$ for each bin j.
- Pre-assignment: $x_{0,0} = 1$ (Item 0 pre-assigned to Bin 0).
- Binary Constraints: $x_{ij}, y_j \in \{0, 1\}$.

Brute Force Approach

Algorithm Steps:

1. Initialize the minimum number of bins $Y = 1$.
2. Assign items to bins while checking capacity constraints.
3. Increment Y until a valid configuration is found.
4. Output the assignment matrix showing which item is in which bin.

Advantages:

- Simplifies the search space by minimizing the initial bin count.
- Exhaustively explores all valid configurations.

Example Problem

Input:

- Item weights: $[3, 2, 3, 4]$
- Bin capacity: $C = 6$

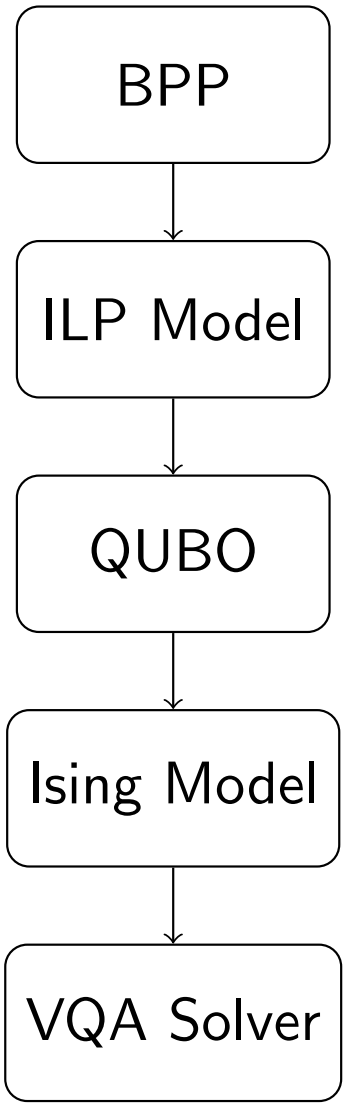
Output:

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

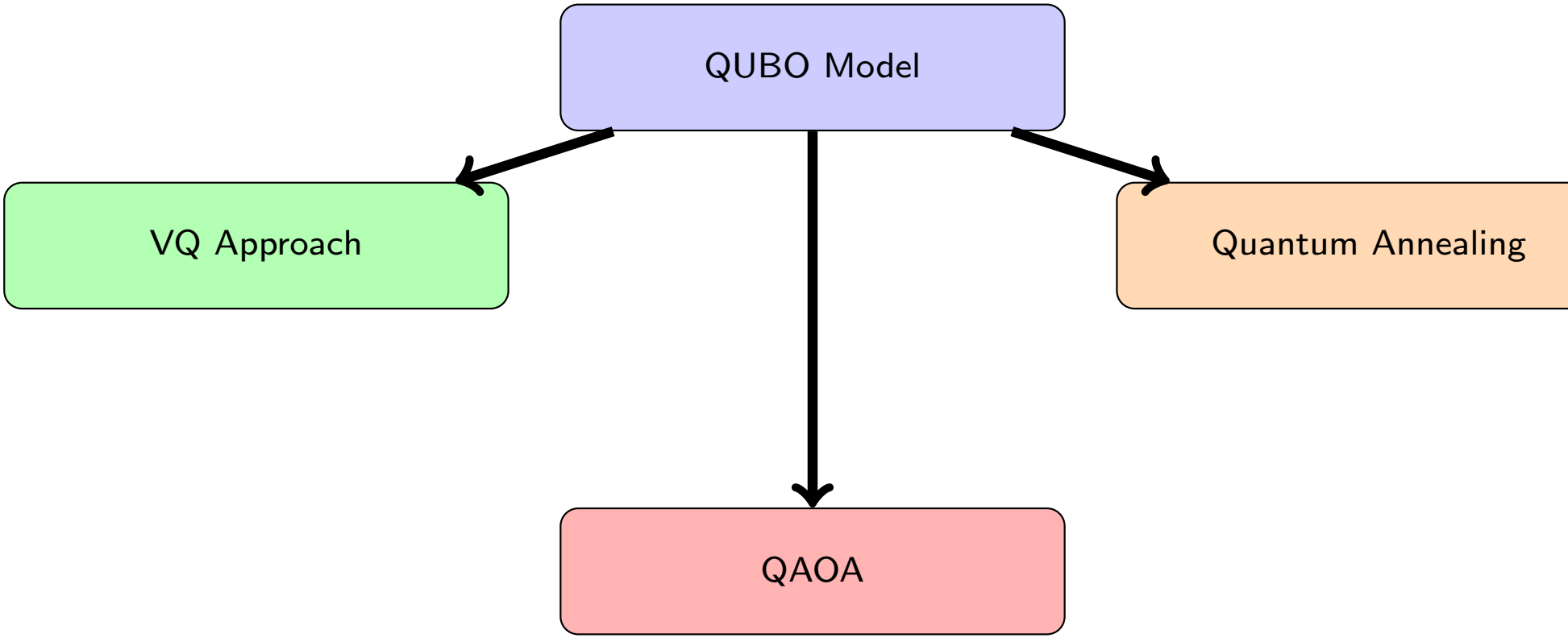
Interpretation:

- Items are packed into 2 bins.
- Bin 1: Items 2, 4
- Bin 2: Items 1, 3

Flowchart: BPP to VQA Solver



QUBO Model and Quantum Approaches



ILP to QUBO Conversion: Overview

- **Objective Reformulation:** Retain the objective of minimizing the number of bins used, i.e.,

$$\sum_{i=1}^{m} y_i.$$

- **Handling Capacity Constraints:** Introduce penalty terms to ensure that bin capacities are respected:

$$(Cy_i - \sum_{j=1}^{n} w_j x_{ij})^2.$$

- **Assignment Constraint:** Ensure each item is assigned to exactly one bin:

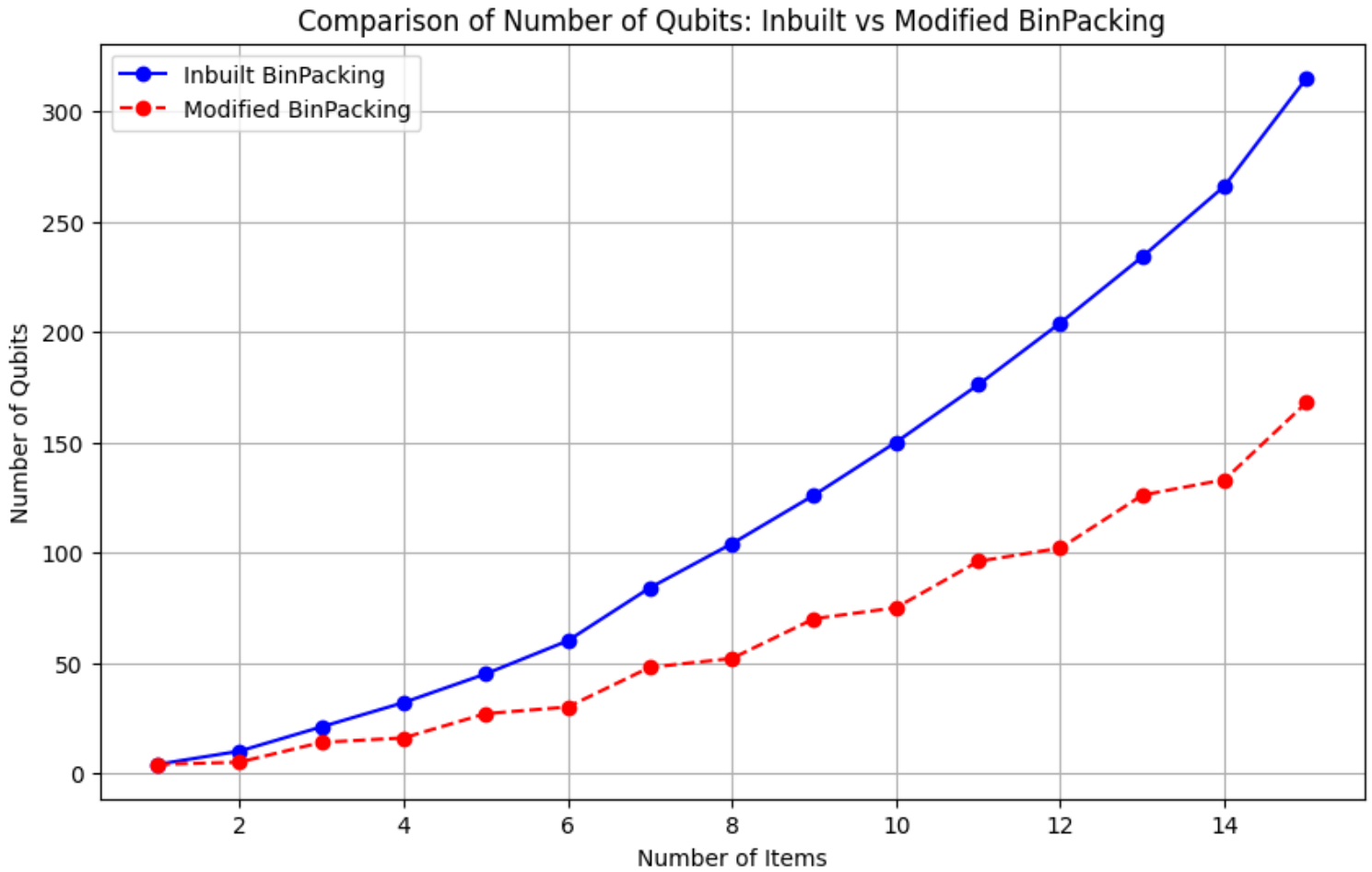
$$\sum_{i=1}^{m} x_{ij} = 1 \quad \Rightarrow \quad P_{\text{assign}} = \beta \sum_{j=1}^{n} \left(1 - \sum_{i=1}^{m} x_{ij}\right)^2.$$

Final QUBO Formulation

The final QUBO objective combines the original objective and penalty terms for capacity and assignment constraints:

$$Q(x,y) = \sum_{i=1}^{m} y_i + \alpha \sum_{i=1}^{m} \left(C y_i - \sum_{j=1}^{n} w_j x_{ij} \right)^2 + \beta \sum_{j=1}^{n} \left(1 - \sum_{i=1}^{m} x_{ij} \right)^2$$

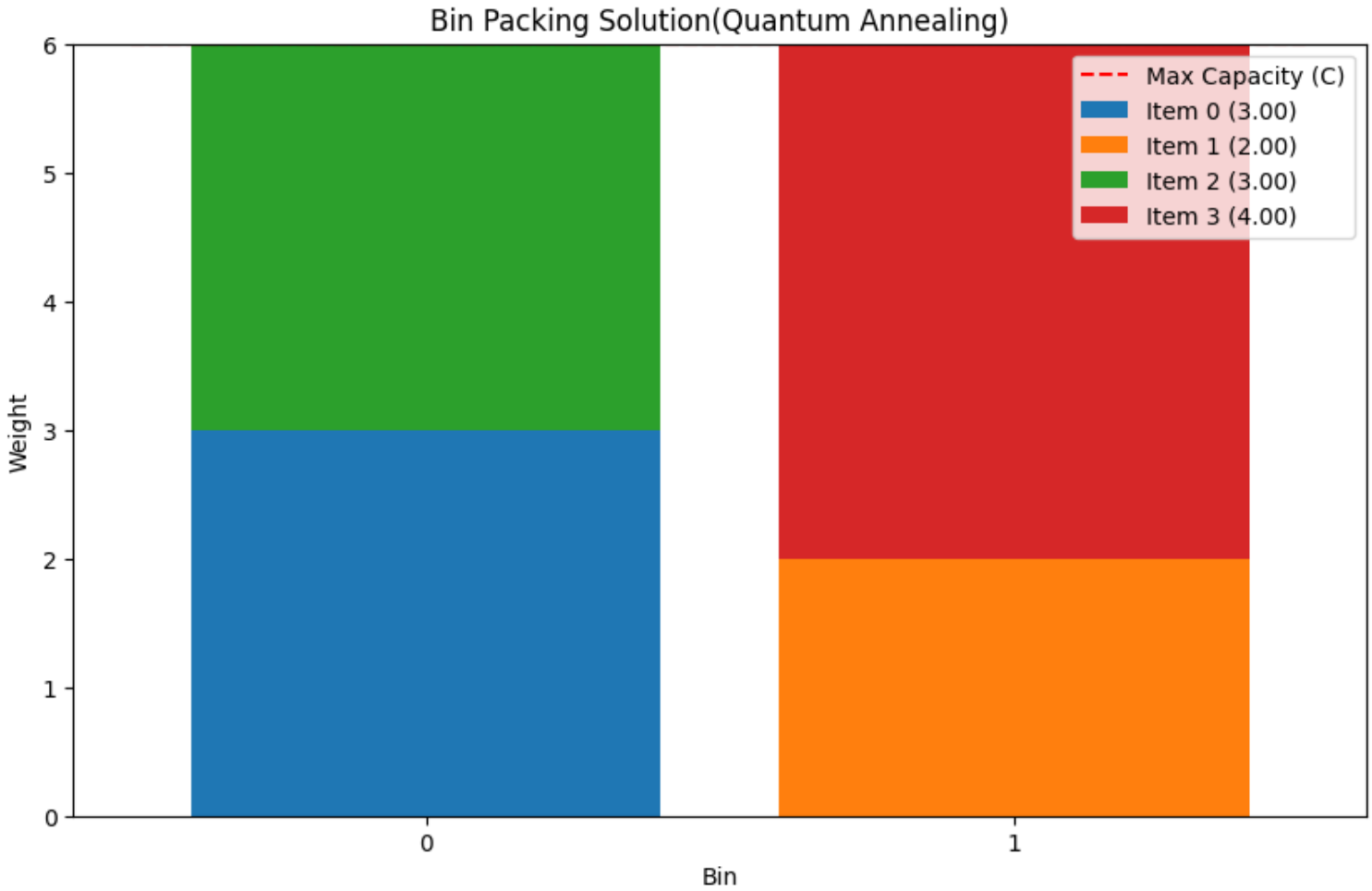
Comparing the number of qubits



Solving QUBO using Quantum Annealing on D-Wave

1. **Initialize the Sampler:**
   - Create an embedding-aware sampler (`EmbeddingComposite`), which wraps `DWaveSampler`.
   - This handles mapping logical qubits of the QUBO to physical qubits on the D-Wave hardware.
2. **Run the Sampler:**
   - Execute the QUBO model using `sampler.sample_qubo()` with 100 reads (`num_reads=100`).
   - Generate multiple samples, each representing a possible solution.
3. **Extract the Best Solution:**
   - Identify the solution with the lowest energy from the samples as the best solution.
   - Extract the solution variables (`best_solution`) and corresponding energy (`best_energy`).

Bar Plot(QA)


Bin Packing Solution(Quantum Annealing)
Max Capacity (C)
Item 0 (3.00)
Item 1 (2.00)
Item 2 (3.00)
Item 3 (4.00)
Weight
Bin
0
1
2
3
4
5
6
0
1

VQA Approach: Overview

1. **Problem Formulation:**
   - Map the optimization problem to an Ising Hamiltonian:

$$H = \sum_i h_i Z_i + \sum_{i<j} J_{ij} Z_i Z_j$$

     where Z_i are Pauli-Z operators, and h_i, J_{ij} are problem-specific coefficients.
   - Goal: Minimize the energy of the Hamiltonian, corresponding to the ground state.
2. **Parameter Initialization:**
   - Design a quantum circuit with parameterized gates (e.g., rotations, entangling gates).
   - Initialize parameters $\vec{\theta}$ randomly to prepare a trial quantum state $|\psi(\vec{\theta})\rangle$.

VQA Approach: Optimization & Solution

3. Energy Measurement:

- Measure the energy expectation value:

$$\langle H \rangle = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle$$

- Evaluate the energy of the trial quantum state on the quantum hardware.

4. Classical Optimization:

- Use a classical optimizer (e.g., gradient-based methods) to update parameters $\vec{\theta}$.
- Minimize the energy expectation $\langle H \rangle$ iteratively.

5. Iterative Refinement & 6. Solution Interpretation:

- Repeat steps 3 and 4 until the energy converges to a minimum.
- Measure the final quantum state $|\psi(\vec{\theta}^*)\rangle$ to obtain the solution to the optimization problem.

Identify the Ordering of Variables

In the QUBO Model,The list of binary variables look like the following:

$x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}, y_0, y_1,$ capacity_bin_0_@int_slack_@0, . . . ,

Each binary variable corresponds to a specific bit in the binary string.

Map the Eigenstate to Variables

Once we have an eigenstate, such as:

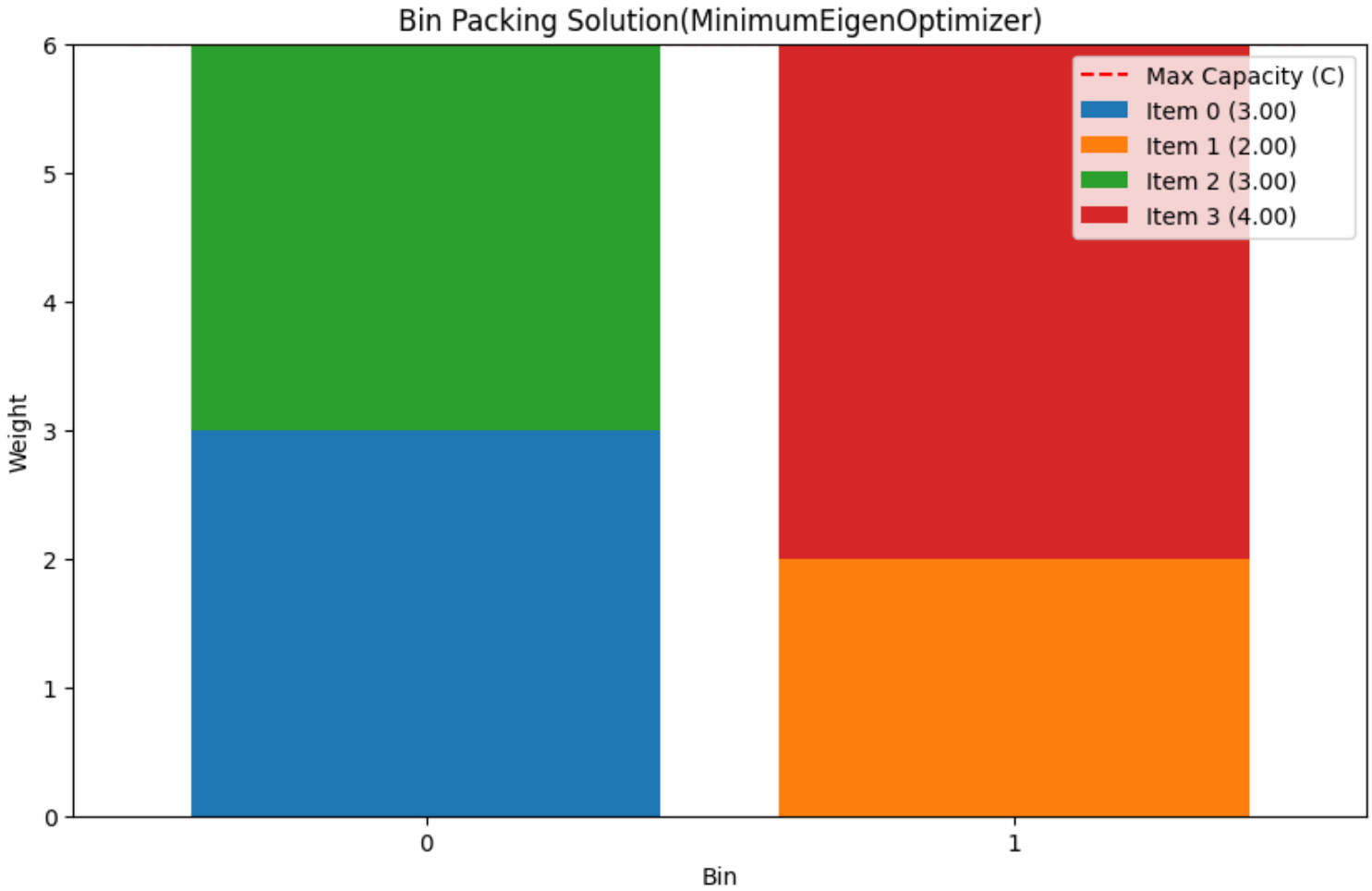
$$\text{eigenstate} = [1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0]$$

We can map these bits back to the variables as follows:

Mapping Binary Variables to Eigenstate Values

Binary Variable	Eigenstate (0/1)
$x_{0,0}$	1
$x_{0,1}$	0
$x_{1,0}$	0
$x_{1,1}$	1
$x_{2,0}$	1
$x_{2,1}$	0
$x_{3,0}$	0
$x_{3,1}$	1
y_0	1
y_1	1
capacity_bin_0_@int_slack_@0	0
capacity_bin_0_@int_slack_@1	0
capacity_bin_1_@int_slack_@0	0
capacity_bin_1_@int_slack_@1	0
capacity_bin_2_@int_slack_@0	0
capacity_bin_2_@int_slack_@1	0

Bar Plot(VQE)


Bin Packing Solution(MinimumEigenOptimizer)
Max Capacity (C)
Item 0 (3.00)
Item 1 (2.00)
Item 2 (3.00)
Item 3 (4.00)
Weight
Bin
0
1
2
3
4
5
6
0
1

Comparison of Approaches

- **Quantum Annealing:**
  - Guaranteed optimal solution.
  - Efficient for small to medium-sized problems.
- **Brute Force:**
  - Simpler to implement.
  - Computationally expensive for larger problems.