Bin Packing Problem (BPP) Formulation and Solution

1 Problem Definition

The *Bin Packing Problem (BPP)* involves optimally placing a set of objects of varying sizes into containers (or "bins") of fixed capacity, minimizing the number of containers used. This problem has applications in logistics, warehouse management, and resource allocation.

Given:

- A set of n objects, each with size s_i where $i = 1, 2, \ldots, n$.
- A set of bins, each with a fixed capacity C.

The objective is to find the minimum number of bins required to accommodate all objects, ensuring that each bin's total object size does not exceed C.

2 Mathematical Formulation

Define:

- $y_j \in \{0, 1\}$: A binary variable indicating whether bin j is used.
- $x_{ij} \in \{0, 1\}$: A binary variable that indicates whether object i is placed in bin j.

2.1 Objective Function

The objective is to minimize the total number of bins used:

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

where m is an upper bound on the number of bins.

2.2 Constraints

The problem is subject to the following constraints:

2.2.1 Object Assignment Constraint

Each object i must be assigned to exactly one bin:

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i = 1, 2, \ldots, n$$

2.2.2 Bin Capacity Constraint

The total size of objects assigned to each bin cannot exceed its capacity:

$$\sum_{i=1}^{n} s_i \cdot x_{ij} \leq C \cdot y_j, \quad \forall j = 1, 2, \ldots, m$$

3 QUBO Conversion

After defining the ILP constraints and objective function, we can convert it to a Quadratic Unconstrained Binary Optimization (QUBO) model. The code for each version is provided below:

3.1 Qiskit QUBO Conversion

To convert the ILP to QUBO format in Qiskit, we use the `quadratic_program_to_qubo` function to apply the appropriate transformations to the Quadratic Program (QP).

3.2 D-Wave QUBO Definition

In the D-Wave implementation, we use the BQM (Binary Quadratic Model) representation, directly defining linear and quadratic coefficients to represent the objective function and constraints.

4 Solution Workflow

To solve the BPP using this approach:

1. Define object sizes and container capacity.
2. Choose either Qiskit or D-Wave implementation.
3. For Qiskit, use the ILP model and convert it to QUBO.
4. For D-Wave, directly define the BQM for the problem.
5. Solve the QUBO model with a suitable optimization method (e.g., quantum annealing).

This document provides a structured approach to understanding and implementing the Bin Packing Problem using both Qiskit and D-Wave for quantum and classical solutions.

5 Create a Brute Force Solver for the QUBO Problem

In this section, a **Brute Force Solver** for the QUBO problem is developed. QUBO, or Quadratic Unconstrained Binary Optimization, is an NP-hard problem where we aim to minimize an objective function expressed in the form:

$$x^T Q x$$

where Q is a matrix of coefficients, and x is a binary vector. The brute force approach involves generating all possible combinations of binary values for x and evaluating the objective function for each configuration. This method is computationally feasible only for small instances, as the number of possibilities grows exponentially with the number of binary variables.

5.1 Steps for the Brute Force Solver

1. **Generate All Possible Solutions**: For a QUBO problem with n binary variables, there are 2^n possible configurations of x.
2. **Evaluate Objective Function**: For each configuration, compute the value of $x^T Q x$.
3. **Select Optimal Solution**: Identify the configuration with the minimum objective function value.

6 Solve the QUBO Using Quantum Annealing Simulators

Quantum Annealing is a quantum-inspired optimization method that solves problems by gradually "annealing" towards the minimum energy state of a system. This approach is well-suited for QUBO problems and can often solve them faster than classical methods by leveraging quantum properties like tunneling.

6.1 Using D-Wave's Ocean Framework

1. **Define the QUBO**: Express the problem as a binary quadratic model, where the terms include linear coefficients (representing individual variables) and quadratic coefficients (representing interactions between pairs of variables).
2. **Choose a Quantum Annealing Simulator**: D-Wave provides simulators like **Simulated Annealing** or **Quantum Annealing** on actual hardware.
3. **Optimize and Retrieve Results**: Run the annealing process and retrieve the configuration that minimizes the objective function.

This approach is beneficial for larger QUBO problems that are infeasible to solve using brute force.

7 Quantum Variational Solver for the QUBO

A **Quantum Variational Approach** uses a parameterized quantum circuit (or **Ansatz**) to iteratively optimize the objective function by adjusting circuit parameters until convergence. This approach is particularly useful when hardware constraints make it difficult to run large circuits.

7.1 Steps to Implement

1. **Define Multiple Ansätze**: Create different trial wave functions (Ansätze) to explore the solution space. Examples include:
   - **Hardware-Efficient Ansatz**: Uses circuits that are efficient on quantum hardware.
   - **Layered Ansatz**: Increases the number of circuit layers for more flexibility.
2. **Hybrid Quantum-Classical Optimization**: Build a hybrid function that uses classical optimizers to adjust the parameters of the quantum circuit, minimizing the QUBO.
3. **Solve the QUBO**: Use the best Ansatz to iteratively converge towards the optimal solution.

8 Quantum Approximate Optimization Algorithm (QAOA) for QUBO

QAOA is a variational quantum algorithm tailored to solve combinatorial optimization problems, including QUBO. The algorithm alternates between two parameterized operators — a "mixer" operator that explores the solution space and a "phase separator" operator that encodes the problem constraints.

8.1 Implementation Process

1. **Define QAOA Ansatz**: Set up the QAOA circuit with alternating layers of mixers and phase separators.
2. **Optimize Parameters**: Use a classical optimizer to adjust the parameters for each layer until the expectation value of the objective function is minimized.
3. **Extract Solution**: The final parameter values give a binary solution to the QUBO that approximates the optimal configuration.

QAOA is particularly useful as it balances the need for a strong quantum solution with feasible circuit depths, making it ideal for QUBO problems on near-term quantum devices.

9 Comparison of QAOA, Quantum Annealing, and Quantum Variational Approaches

In the context of solving optimization problems, especially QUBO problems, three prominent quantum methodologies have emerged: Quantum Approximate Optimization Algorithm (QAOA), Quantum Annealing, and Quantum Variational approaches using different Ansätze. Understanding the differences among these approaches and their performance against classical methods, such as brute force, is essential for evaluating their efficacy.

9.1 Differences Between Approaches

- **Quantum Annealing:** This method utilizes quantum mechanics to find the minimum energy state of a system, effectively solving optimization problems by exploiting quantum tunneling. It is particularly well-suited for problems like QUBO, as it can quickly navigate complex energy landscapes. D-Wave's quantum hardware is a prominent example that implements this method.
- **QAOA:** This is a hybrid quantum-classical algorithm that combines quantum circuits with classical optimization. QAOA involves alternating layers of quantum gates to represent the problem constraints and a mixer operation to explore the solution space. It allows for a tunable depth of circuits, which can be beneficial for near-term quantum devices.
- **Quantum Variational Approaches:** These methods leverage parameterized quantum circuits (Ansätze) to approximate the ground state of a quantum system. Variational methods typically involve classical optimization of the circuit parameters and can employ different types of Ansätze, which may significantly affect performance and convergence.

9.2 Performance Comparison with Brute Force

After conducting experiments across various problem sizes, the following findings emerged regarding the performance of brute force versus quantum approaches:

- **Brute Force Method:** This approach excels in small instances of the Bin Packing Problem (BPP). It can effectively solve instances with 1 to 14 binary variables in under 30 seconds. However, once the problem size exceeds 16 binary variables, the brute force method becomes infeasible on a standard CPU, such as the Mac M3.
- **Quantum Annealing:** Using a real quantum computer, quantum annealing demonstrated promising performance: small instances were solved in approximately 0.9 seconds, medium instances in about 2 seconds, and larger instances took around 17 seconds. While fast, there were instances where the solutions provided did not always meet all constraints.
- **Quantum Variational Circuits (Pennylane):** When attempting to solve small instances, the variational circuit showed potential but faced challenges. The cost function reached a value of 0, but the optimization process struggled to progress further, likely due to difficulties in calculating gradients with certain function approximations (e.g., using a tanh function). The trials took about 12-20 minutes without achieving optimal solutions, indicating potential hardware limitations.
- **QAOA Performance (using Pennylane and Qiskit):** Similar results were observed with QAOA implementations, where the circuit often stalled close to a cost function value of 0. Despite the theoretical advantages of QAOA, practical implementations on classical simulators resulted in long computation times without yielding good solutions.

In conclusion, while brute force remains the most effective for small problems, quantum methods like quantum annealing appear more efficient for medium to large instances. The variational approaches, though promising, need further optimization and potential real quantum hardware utilization to fully realize their benefits.