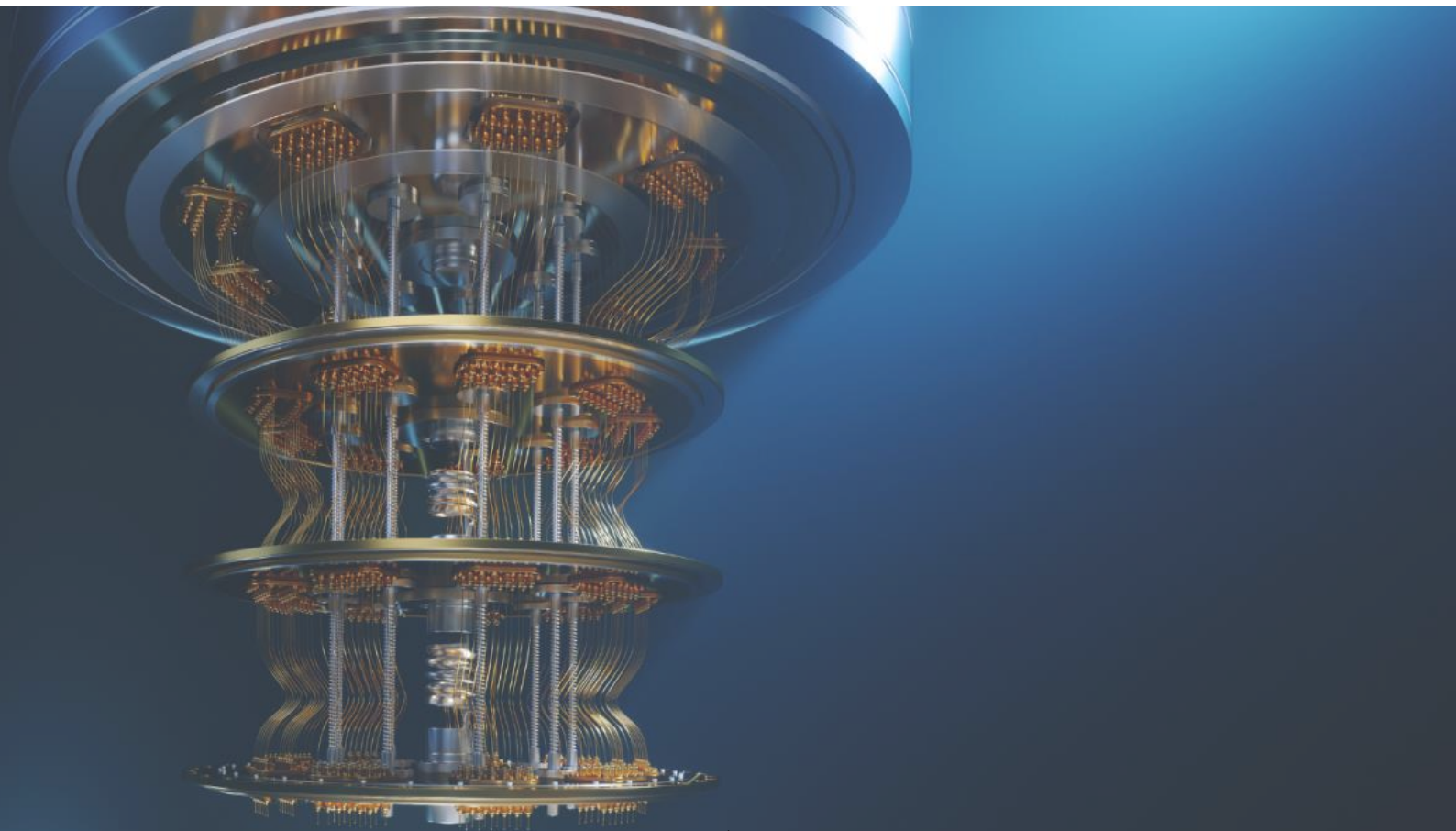

October 10, 2024

Bin Packing Problem (BPP)

Name	Country
Hussein Shiri	Lebanon

The BPP problem requires us to be able to distribute packets or boxes having specific weight or size on different bins or trucks having also maximum weight or size with the requirement of minimizing or decreasing the number of bins or trucks used. In this task weight will be used instead of size.

1 Approaches Used

Inplementing an optimization solution for this problem can be done in several ways:

- Variational Approach - VQE
- QAOA
- Dwave - ExactSolver - BruteForce
- Dwave - Annealer

In the next sections, the approaches used will be explained with some results.

2 Preparing Model Equations

Before stating what equations are being prepared or used, it is worth mentioning that we are preparing two QUBO(Quantum Unconstrained Binary Optimization) models:

- QUBO model for Dwave
- Qubo model for cplex/qiskit

The following equations are implemented in the Problem() class.

- minimize $\sum_{i=1}^{u} y_i$
  This is the objective function, in which we want to minimize the number of used bins $x's$ or trucks $y's$
- $\sum_{j=1}^{n} w_j x_{i,j} \leq c y_i$
  The total weight of the packets in a bin should be less than the maximum weight the bin can hold.
- $\sum_{i=1}^{u} x_{i,j} = 1$
  This equation checks that every packet is used and only found once, for example the packet cannot be used more than once in same bin or different bins.

3 Using VQE

In the first approach we use VQE (variational solution). To use VQE, we perform the following steps:

- Transform QUBO model to ising model
- Find the number of qubits from the ising model
- Implement an ansatz from the number of qubits
- Run SamplingVQE to find the values of x's and y's.

```
vqe_optimizer = MinimumEigenOptimizer(vqe)
vqe_result =vqe_optimizer.solve(
self.qubo_vqe_qaoa)
return vqe_result
```

Fig 1: Part of the code used to run VQE

In VQE 3 types of TWOLOCAL ansatz have been compared, with each having 3 qubits:

- entanglement:linear, cz, ry gates

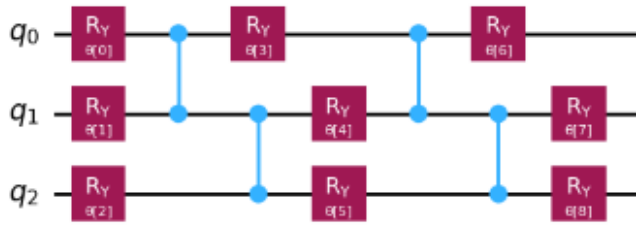

Fig 2: ansatz1

- entanglement:linear, cx, ry gates

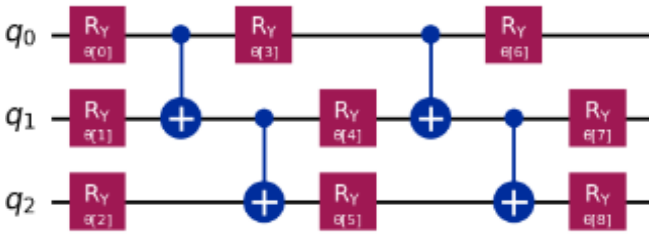

Fig 3: ansatz2

- entanglement:full, cz, ry gates

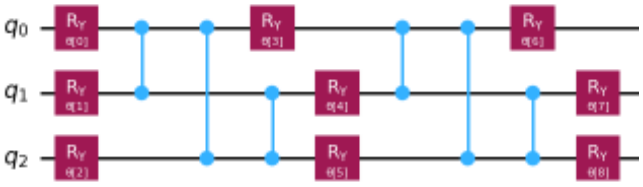

Fig 4: ansatz3

3.1 Used arguments

- Number of trucks/bins = 3
- Number of packets/boxes = 2
- Weight of packets = 1, 2
- Max weight per truck/bin = 5

VQE(cz,ry,linear)	VQE(cx,ry,linear)	VQE(cz,ry,full)	QAOA	Bruteforce	Annealing
561.65	624.90	794.33	567.62	31.35	0.09328
Good	Good	Excellent	Good	Tries all solutions	Several solutions

Table 1: table showing the time(seconds) and results for different techniques

Although this looks a small problem, i was not able to run problems with higher numbers due to memory size when the number of variables and equations increase.

```
MemoryError: Unable to allocate 4.00 GiB for an array with shape (2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2) and data type complex128
```

Fig 5: example of error got when increasing number of trucks or packets

3.2 VQE results

The ansatz having full entanglement (ansatz3) was the one that gave the best results. Since we have maximum weight per bin = 5 and we have 2 packets with weights 1 and 2 respectively so we need only 1 bin. Ansatz3 was the only one able to use only 1 bin in the solution. It is also worth mentioning that this was done using COBYLA with max iteration of 30.

4 QAOA

QAOA is better for working on combinatorial problems while VQE is better for working on chemistry calculations.

To work with VQE in this problem we first need to define an ansatz but for QAOA we do not need to define an ansatz.

Also VQE ansatz with full entanglement was able to surpass QAOA since QAOA used 2 bins and not 1.

5 Dwave

Dwave has several solvers, 2 solvers will be used, one for exact solving (brute force) and one for solving the problem using quantum annealing.

5.1 Brute force

We can implement brute force using the Dwave ExactSolver, this solver tries all solutions. It is still an open question how to get the best solution from all tried solutions since this solver returns all tried solutions.

5.2 Annealing approach

It is worth mentioning that this technique produces not one result but several results. And also it took the least time.

```
result = QBSolv().sample_qubo(self.
        qubo_dwave[0])
return result
```

Fig 5: Code used to run annealing solution

6 Limitations and future work

Of course this solution can be expanded more and it had some limitations in it that stood in front of the ability to expand, some limitations are:

- Computational resources
- Dwave had very few tutorials for QUBO, most being old.
- It is required to implement a different QUBO model for different quantum programming languages.

On the other side, we can expand this project in the following ways:

- Using HPC(high performance computing)
- Trying optimization for other problems
- Trying more different ansatz