Exploring QAOA via the Max-Cut Problem

Cyrus Majd

PART 1

Questions 1,2, and 3:

The QAOA algorithm is an example of a variational algorithm that can be run on NISQ quantum machines. In this project, we explore the application of QAOA in the classic Max-Cut problem. To understand the Max-Cut problem, first imagine a set of nodes in a plane. Now, we have to chain all of these nodes with edges. Each node must have at least one edge. To visualize, see the below image depicting an example graph of 10 nodes, with each node having at least 1 edge:

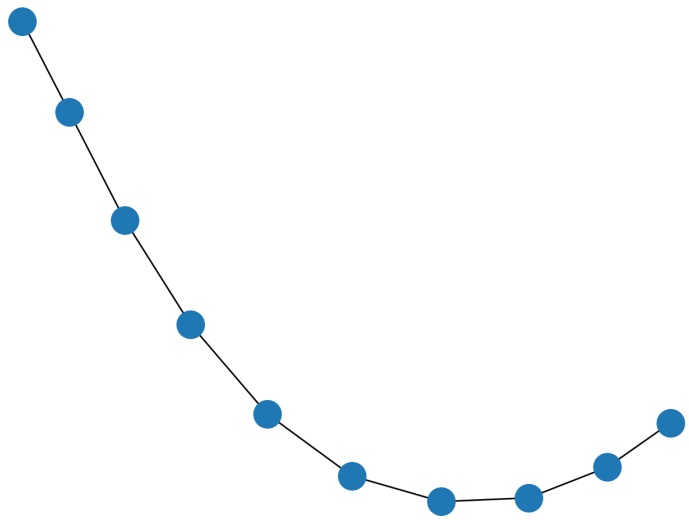
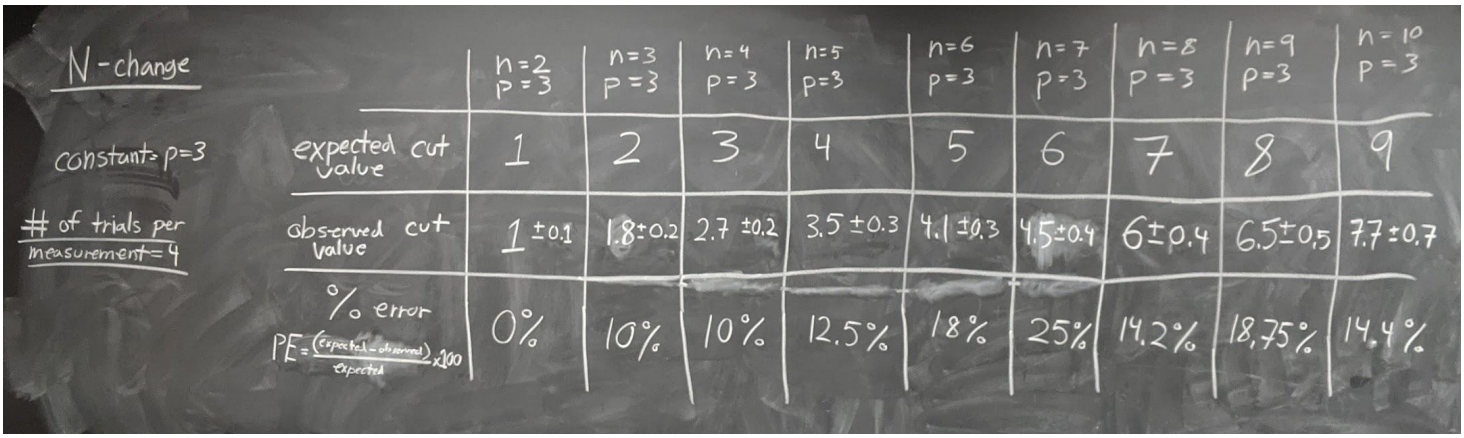


Figure 1: A graph of 10 nodes. This kind of graph is used for the max-cut problem. Your graph could have more edges per node than this one too.

If we are given the freedom to move any and all of these nodes to wherever on the plane we want, and if we can imagine having an infinitely large and thin knife, one can imagine that it would be possible to cut through all of the edges in this graph in a single slice! This is the guiding principle of max-cut. The question now becomes if our QAOA algorithm can correctly calculate the maximum number of edges we can slice with one cut.

Utilizing the given QAOA code provided here, we can experiment with this problem by changing the values of N and P. To keep our experiments controlled, we will change one parameter at a time. First, let's change the value of N [the number of nodes].

By increasing the value of N, we should first expect that the computation time of the program will increase. We also should expect the program to give us a higher max-cut value. The below chart depicts the observed max-cut value from the simulation compared to the actual max-cut value, for increasing N values, a constant P value of 3, and where each experimental data point is the average of 4 trials.



N-change	n=2 p=3	n=3 p=3	n=4 p=3	n=5 p=3	n=6 p=3	n=7 p=3	n=8 p=3	n=9 p=3	n=10 p=3
expected cut value (constant: p=3)	1	2	3	4	5	6	7	8	9
observed cut value (# of trials per measurement = 4)	1 ±0.1	1.8±0.2	2.7 ±0.2	3.5 ±0.3	4.1 ±0.3	4.5±0.4	6±0.4	6.5±0.5	7.7±0.7
% error $PE = \frac{(\text{expected} - \text{observed})}{\text{expected}} \times 100$	0%	10%	10%	12.5%	18%	25%	14.2%	18.75%	14.4%

Figure 2: Chart depicting experimental values for changing N and static low P value

The data shown in Figure 2 shows an evident increase in inaccuracy for the value of max-cut as N increases. Surprisingly, the inaccuracy seems to stabilize at around 18.07% after N = 5 [I had expected the inaccuracy to reliably increase with the value of N as it did for the first 5 values of N]. Now while these specific values tell a compelling story, it does not encapsulate the whole truth. I found that the number of layers, P, drastically affects the results of the max-cut algorithm in a way that makes a lot of sense in hindsight.

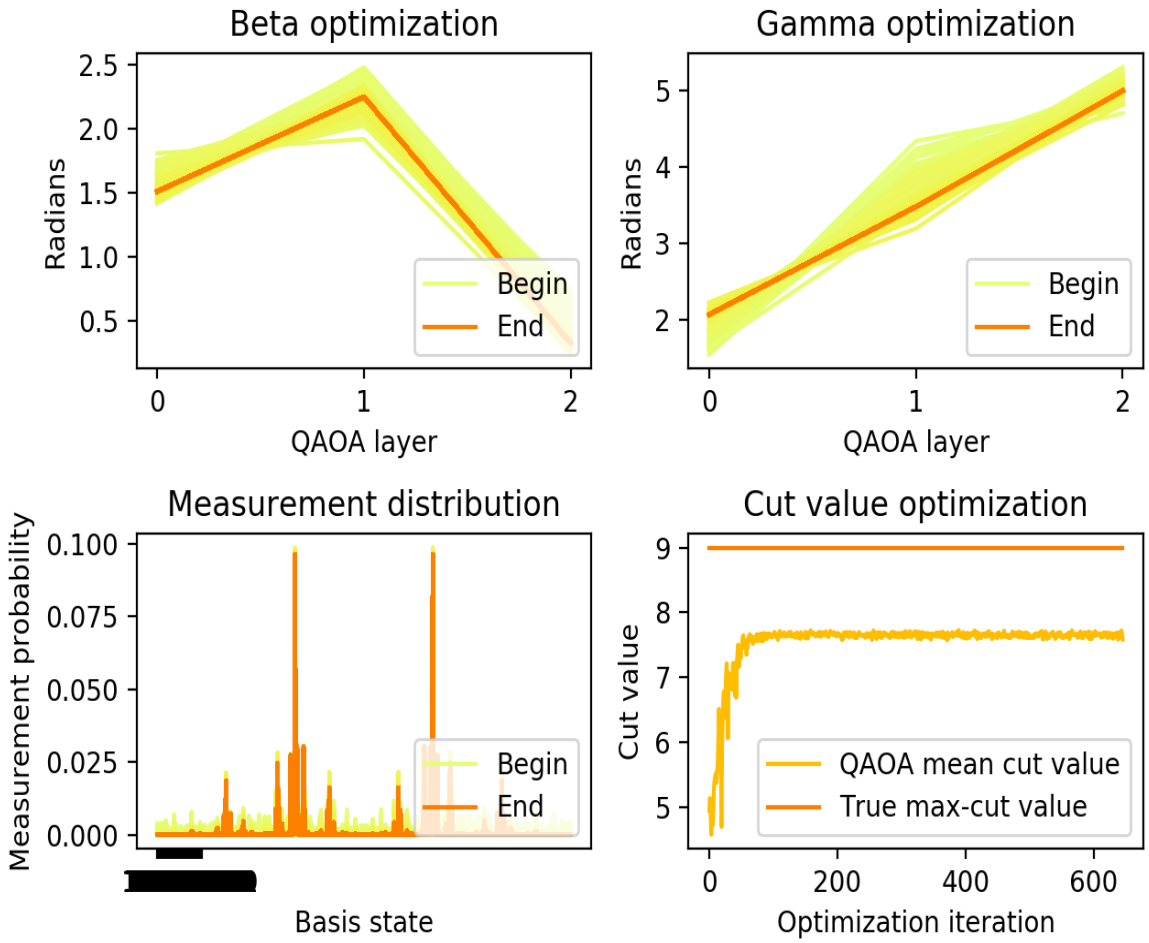


Figure 3: Max-cut results for N = 10 and P = 3

In the experiment above, our P value is stuck at 3. Because the number of layers directly affects the number of chances the algorithm has to fine-tune its prediction, you can see how the Cut Value Optimization graph seems to flatten out very quickly at around < 100 iterations. However, if we increase the value of P with N, we can see a more gradual curve. The figure below will depict the results of the max-cut algorithm with N = 10 and P = 10.

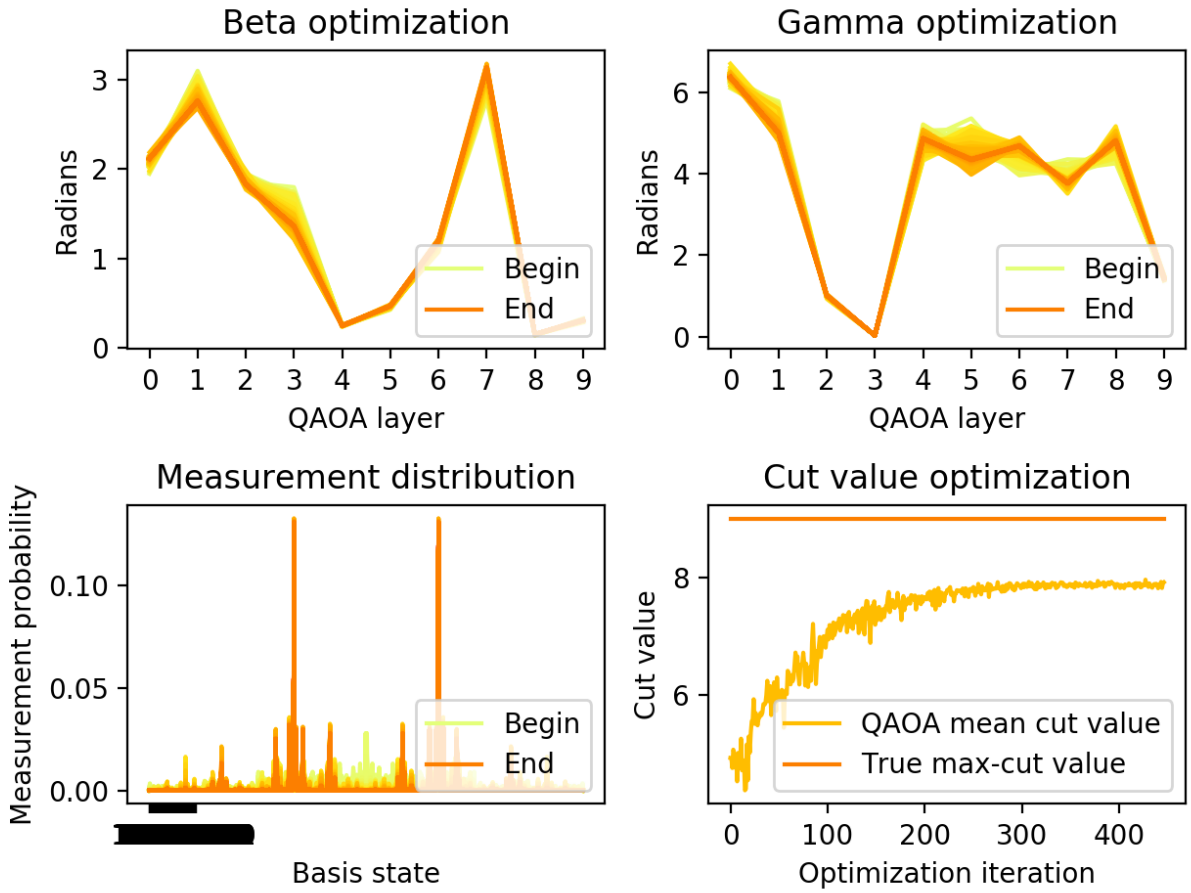


Figure 4: Max-cut results for N = 10 and P = 10

The results show that each layer in the P = 10 experiment is more fine-tuned, hence the more gradual prediction of cut-value in the bottom right graph. So although both P = 3 and P = 10 for N = 10 yields similar final results for the cut-value, it seems that we achieve our final value more quickly (but sometimes less accurately) with a smaller value of P. Increasing the P always increases the accuracy of the prediction but in turn it takes more optimization iterations to get there. The reason why more layers of P makes our prediction more accurate is that we might start with bad values of Beta and Gamma, and with more layers (depth) to the algorithm, the more chances we have to recover from the bad initial values.

But why does the QAOA max cut value never reach the true max cut value? The answer lies among the two parameters we haven't yet toyed with; the "maxiter" and "repetitions" arguments to the main method.

We can visually see that the Cut Value Optimization graph in figure 4 is much more gradual than the same graph in figure 3. We can imagine that if we kept increasing

the value of P with a large value of N, the graph would show an even more gradual ascent. It is possible that the limit on the x-axis (determined by the maxiter parameter) may then be a hindering factor in our quest to reach the true max-cut value. In this scenario, a low maxiter value can contribute to inaccuracy. To illustrate this, consider figures 5 and 6 below. The only difference between these two simulations is the value of maxiter. With a larger value of maxiter, we achieve a higher approximation ratio when the simulation has not yet plateaued at any certain max-cut value. Another interesting observation is that increasing the value of maxiter seems to dampen the amplitude of all the "noisy" answers in the measurement distribution graph. This is probably what allows us to have a more accurate QAOA mean cut value.

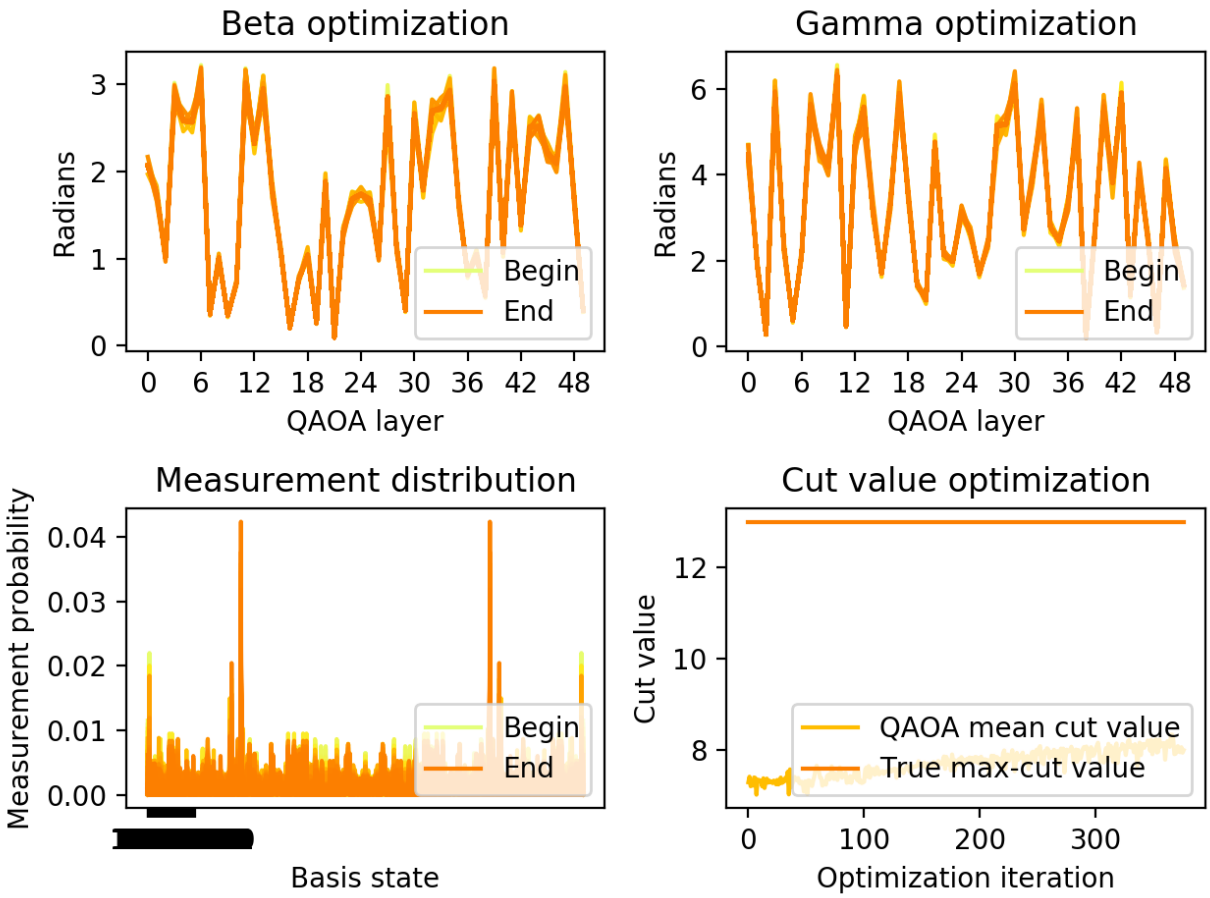


Figure 5: N = 10, P = 50, repetitions = 10,000, maxiter = 256

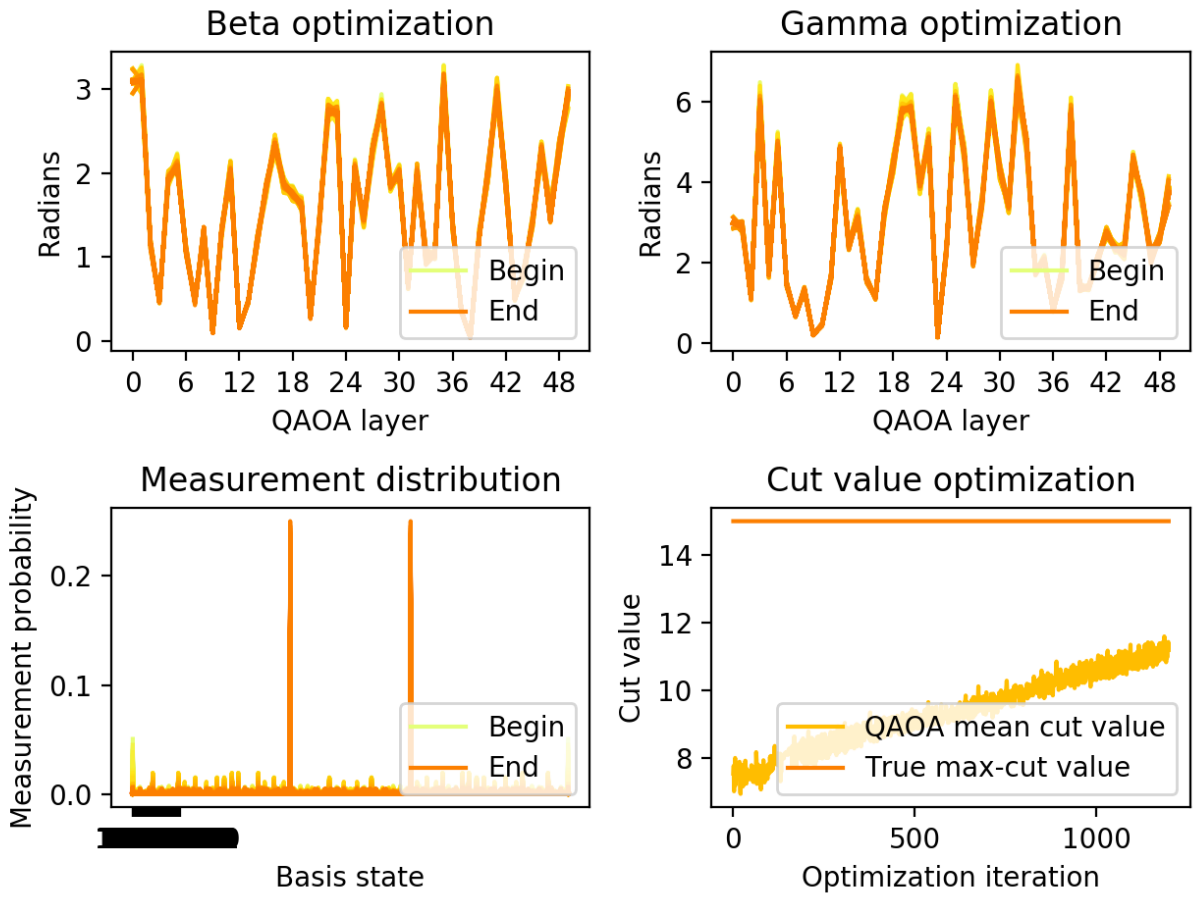


Figure 6: N = 10, P = 50, repetitions 10,000, maxiter = 1000

However, maxiter's value changes pale in significance when compared to the varying results achieved from changing the value of the "repetitions" parameter. "Repetitions" basically controls the number of samples taken from the distribution resulting from the QAOA algorithm. The more samples taken, the more accurate the results will be. The best part is that sampling more points is not nearly as computationally expensive as increasing the value of P, which means that we can get really accurate results with a high repetitions parameter and a relatively low P value. A perfect middle-ground!

Question 4:

Here, we have to recreate the noiseless simulated data in figure 5 of the Arute et al. paper. We can use the vary_p_qaoa.py file and slightly modify it to achieve the results we want to see, but first we have to identify the exact experimental setup used in the figure. First, we have to make sure our value for N is > 10. The value of P also has to

equal 5. So once these parameters are set, all we can do to replicate the diagram in the Arute et al. paper is to change the values of the "repetitions" and "maxiter" parameters, just like how we did in the qaoa.py file. Our results fall in line with what we were getting before as we change the values of these parameters. For example, the figure below shows the approximation value achieved for a repetition value of 10.

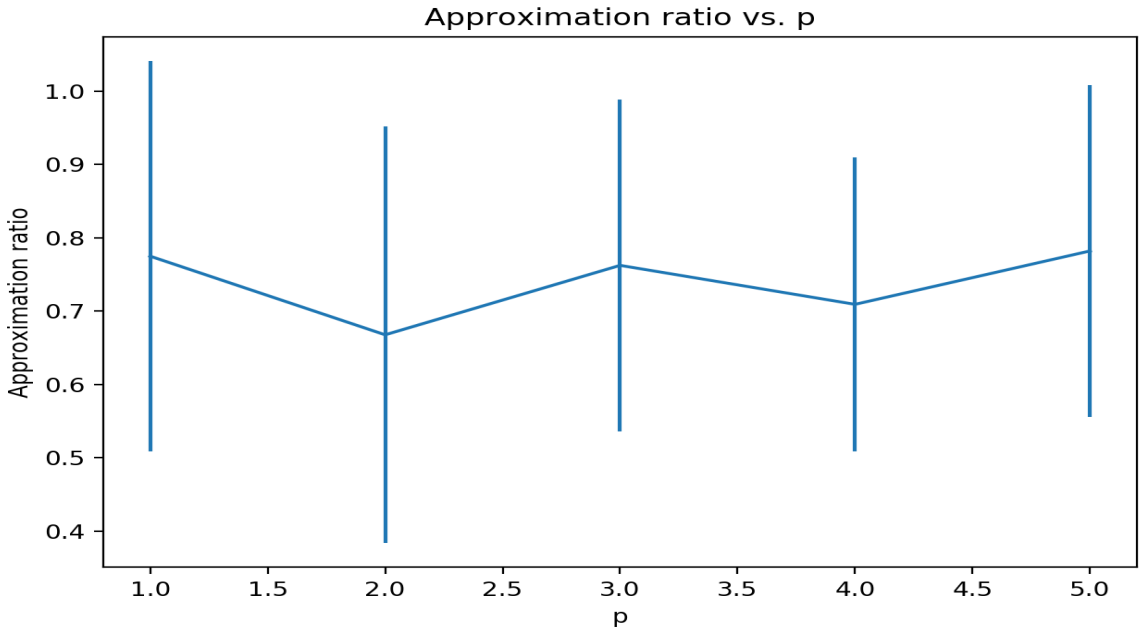


Figure 7: Approximation ratio vs P with P = 5, N = 12, maxiter = 128, and repetitions = 10

If we increase the value of repetitions, we get more and more accurate results. Figure 8 below shows a much less chaotic curve, aligning with what we expect to see in an ideal, noiseless simulation. Figure 9 below shows an even more optimized version, with repetitions = 10000 and an ever decreasing error for increasing values of P.

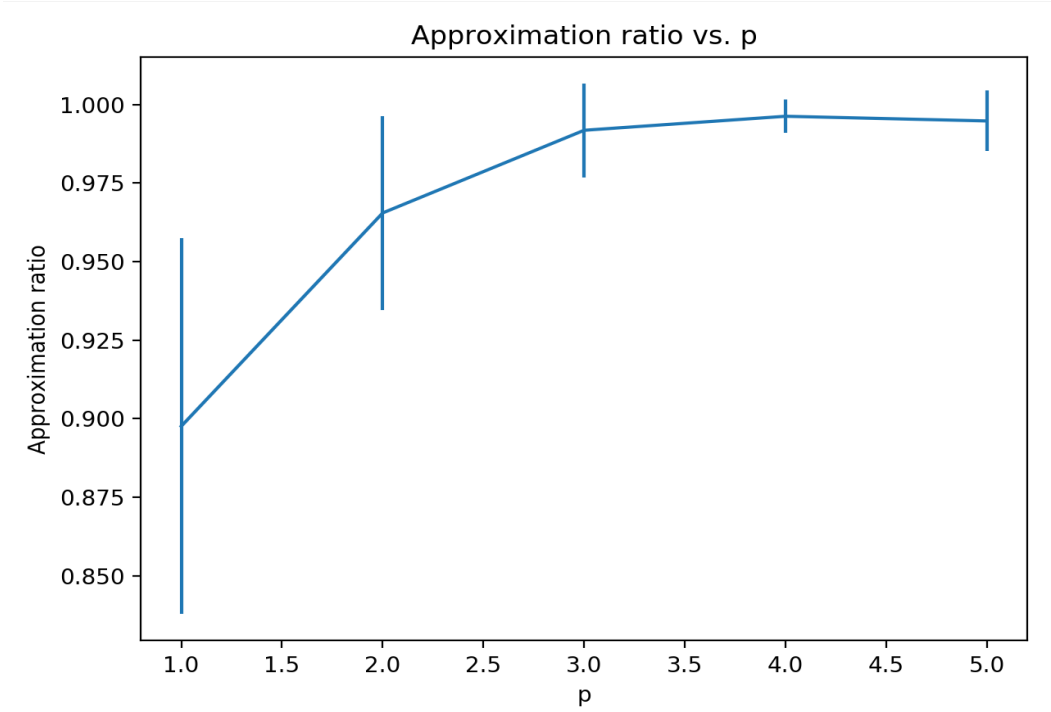


Figure 8: Approximation ratio vs P with P = 5, N = 12, maxiter = 128, and repetitions = 1000

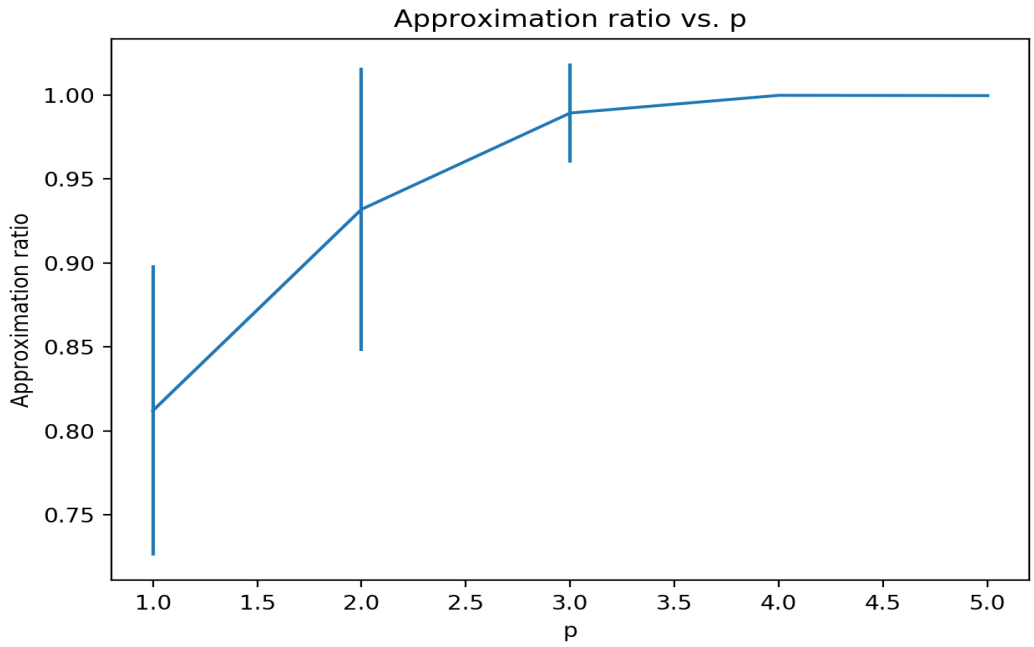


Figure 9: Approximation ratio vs. P with P = 5, N = 12, maxiter = 128, and repetitions = 10000

So in conclusion, we were able to replicate the results of the Arute et al. paper simply by changing the number of repetitions, which increased the accuracy of our ratio. We also found that increasing the value of maxiter helped, but only in cases where N was super large.

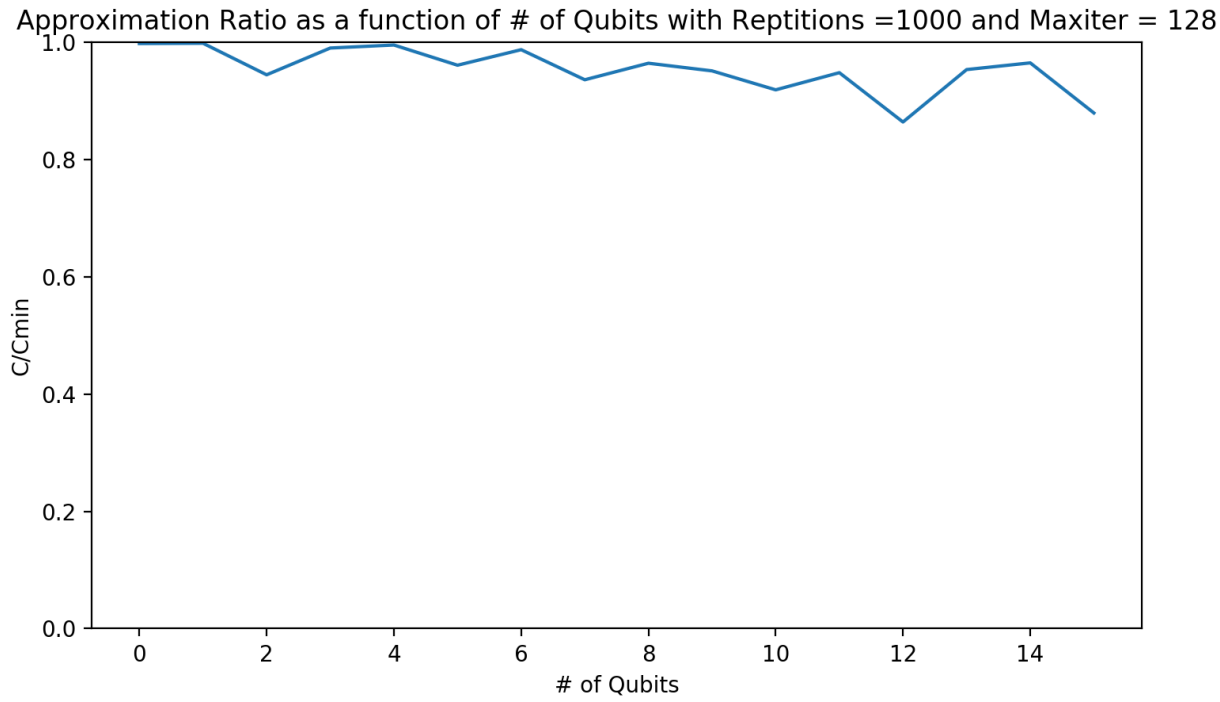
PART 2

Questions 1-5:

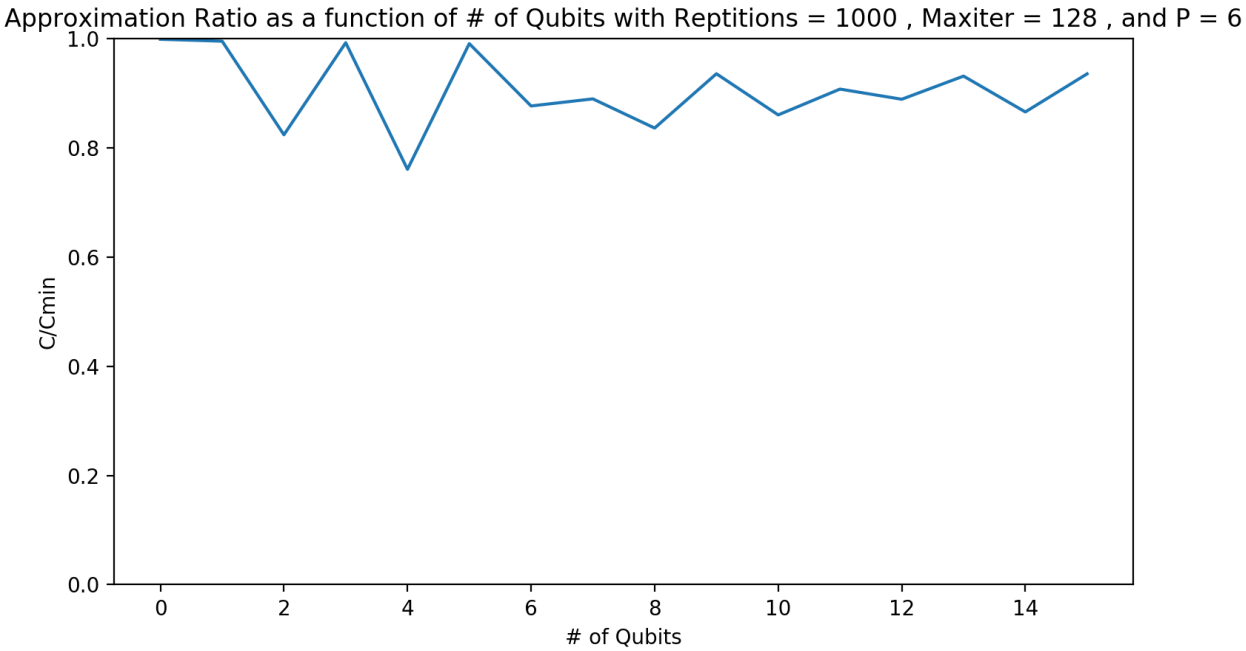
For this part of the assignment, we went ahead with option #1, which was recreating figure 4 in the Arute et al. paper. The objective of this is to basically demonstrate the error of QAOA's optimization efforts

The experiment goal is to demonstrate a decrease in prediction accuracy of the max-cut algorithm for increasing values of N, where N is the number of qubits being simulated in the program. The theory that the accuracy will decrease is supported by two arguments; firstly, that increasing the number of qubits (nodes) in a graph results in more complicated possible cuts the max-cut algorithm could make [thus making it more computationally expensive to find the right cut], and secondly, that the more qubits you introduce into the algorithm, the higher chance there is for error in measurement due to noise. For our simulation, we assume that the algorithm is running on a noiseless, perfect quantum computer, so the second source of error is ignored for our experiment.

To create a contained experiment, we will keep all variables in the QAOA algorithm constant except for the number of qubits, N. This is also necessary to replicate figure 4 in Arute's paper, as the x-axis [control variable] of his graph is in fact the number of qubits run for his experiment. We set the value of repetitions to 1,000, the value of maxiter to 128, and the P value to 10. Our maximum computed N will be 15 and we will calculate the value of max-cut for all values of N from 0 and leading up to 15, as was done in Arute's paper. After running the experiment, we generate the following graph:

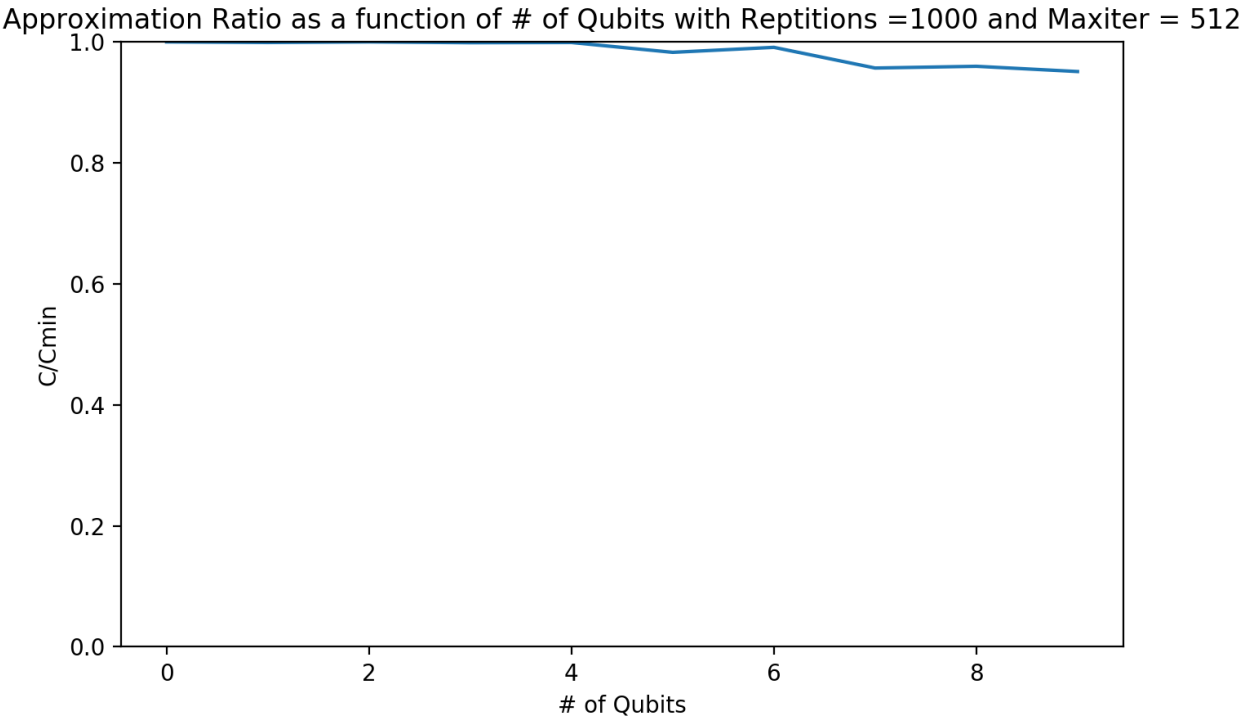


These results do show a decrease in the approximation ratio as a function of the number of qubits, but the descent is rather gradual. To be accurate to Arute's results, we need to have the approximation ratio at least dip below 80%. To make this happen, we can change a few parameters in our simulation. We learned from part 1 that the approximation ratio can decrease with lower values of P and maxiter. A lower repetitions parameter will decrease the number of samples we take from the final measurement distribution, but this just means we will have less accurate results, so we will not lower or raise this parameter. So with this information being known, the simulation is run once again, but this time with approximately half the P value (now 6), repetitions = 1000, and maxiter = 128.

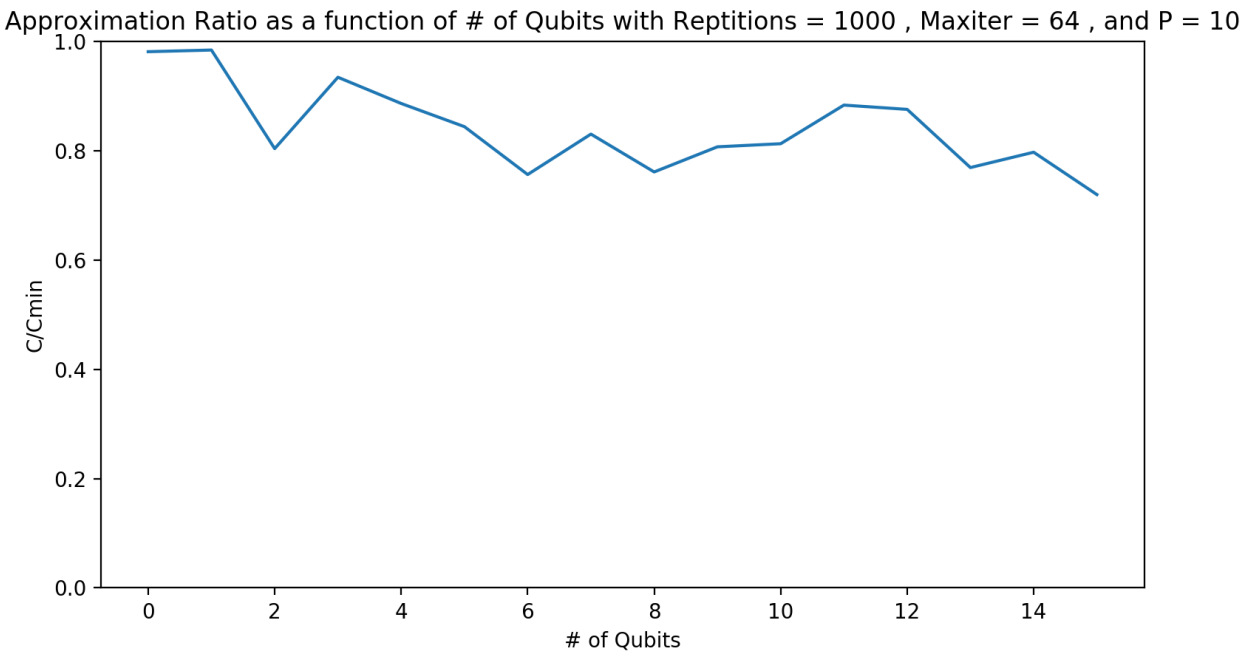


Unfortunately, lowering the P value seems to just make our results more chaotic. This makes sense because if the P value is too low for the algorithm to fully see the graph, we will get unreliable results for our predictions. The error margins increase with a decreased P.

Let's try to mess around with the maxiter parameter. Because maxiter basically increases the number of iterations of the algorithm, we should expect that the greater the value of maxiter, the more accurate our results will be. Sure enough, this is exactly what we see:



The figure above shows results for P = 10 and maxiter = 512. As we can see, the approximation ratio stays pretty high for the majority of the simulation. Of course, this is at the expense of computation time, as maxiter = 512 more than doubles the amount of time the simulation takes. If we lower the value of maxiter, we should expect less accurate results.



The figure above shows our final simulation, one with P = 10, repetitions = 1000, and maxiter = 64. The lower maxiter value and the increasing values of N contributed the most to the trend of decreasing accuracy in the approximation ratio.

Our results greatly resemble the fourth figure in Arute's paper. Our graph shows the approximation ratio (our accuracy for the max-cut value) as a function of the number of qubits per simulation. As we can tell, the accuracy of our max-cut prediction steadily decreases. However, where Arute's graph depicts a trend with a diminishing decrease, our trend seems to have a linear decrease, on average. This could be due to experimental error or if Arute had a different set of initial parameters (maxiter, P value, repetitions value or their analogies) for their experiments. But regardless of slight error, our findings are significant because it actually demonstrates our theory that a more complicated graph results in a more complicated [and often less accurate] max-cut estimation. In addition, it further backs up Arute's findings, which he uses as basis for further theories and simulations. In summary, it teaches us a pretty nice lesson on how increasing the complexity of a problem for QAOA yields much more inaccurate results when we refuse to adjust other variables accordingly.