QUKnights

Alex Georgiev

Quantinuum Challenge

Intro: The QAOA aspect of the Quantinuum challenge provided us with a hybrid quantum-classical optimization strategy to solve the max-cut problem (actually, just get a close approximation for the solution). Our project focuses on improving upon the QAOA algorithm given to us. The purpose of this algorithm is to optimize the mixer and cost angles by creating a circuit with one set of angles, sending it to the quantum computer and back for the result, and using a custom classical optimization algorithm to find the angles that maximize the energy of the Hamiltonian. The reason our project aims to maximize the energy of the Hamiltonian is because the max-cut problem was encoded into quantum gates such that maximizing the Hamiltonian would increase the probability of our quantum computer finding the correct solution.

Initial problem: The provided function for optimizing the mixer and cost angles followed a very naive approach of simply guessing the mixer and cost angles, creating a circuit with them, and if those angles resulted in creating a circuit that calculated a higher energy for the Hamiltonian compared with the previously calculated value, then those angles were stored as the best guess. This process was repeated 1000 times. This method is not very systematic and can’t assure that the maximum energy was truly found. Basically, it simply guesses 1000 parameters for the angles and chooses the best one out of those guesses.

Initial function results: The results of the initial function show that it found the maximum energy to be<FILL IN> and the quantum computer found this solution around 42% of the time. To calculate the percentage of time the quantum computer found the solution, we assume that the solution was the two results that the quantum computer calculated the highest percentage of the time. This is because increasing the energy of the Hamiltonian should increase the probability of finding the correct solution.

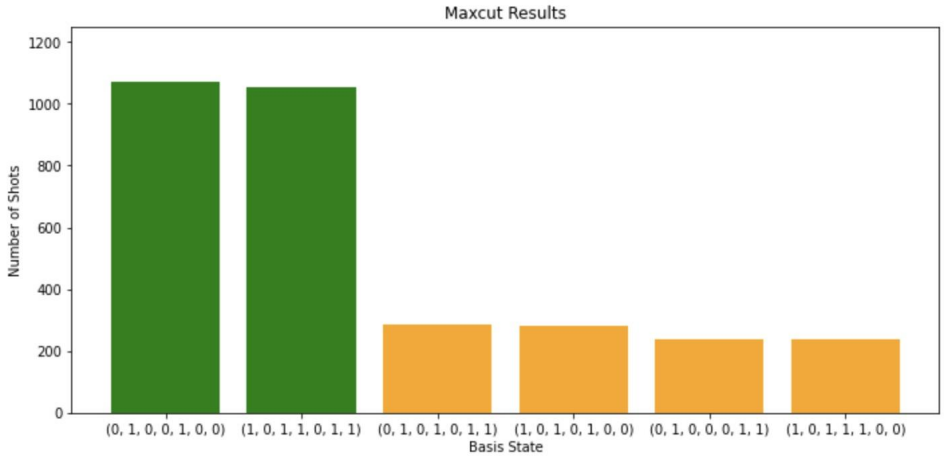
```
%%time
res = qaoa_calculate(backend, backend.default_compilation_pass(1).apply, shots = 5000, iterations = 100, seed=12345)
```

```
new highest energy found:  3.1432
new highest energy found:  3.283599999999999
new highest energy found:  4.361
new highest energy found:  4.925600000000001
new highest energy found:  4.941999999999999
highest energy:  4.941999999999999
best guess mixer angles:  [0.392 0.247 0.138]
best guess cost angles:  [0.592 0.738 0.608]
CPU times: user 34.4 s, sys: 25.5 ms, total: 34.4 s
Wall time: 34.4 s
```

```
from maxcut_plotting import plot_maxcut_results

plot_maxcut_results(res, 6)
```

```
Success ratio 0.4252
```



Solution: My first objective for this challenge was to improve this function such that it could almost guarantee finding the global maximum of the Hamiltonian's energy and therefore increase the probability of the quantum computer finding the correct solution. The process for my algorithm is as follows:

- Guess a set of parameters for the cost and mixer angles
- Slightly increase and decrease the angles by 0.01 (the angles were between 0 and 1)
- There were four possible solutions for each iteration:
    - Increase both the cost and mixer angles
    - Decrease both the cost and mixer angles
    - Increase the cost and decrease the mixer angles
    - Decrease the cost and increase the mixer angles
- The maximum energy calculated from these possibilities was chosen as the local maximum
- If the algorithm found a highest max energy, then it would repeat by continuing to search for solutions in the same direction (i.e. basically following the gradient of the energy function)
- This was repeated 1000 times and each iteration stored a list of local maximums
- The global maximum chosen was the max of the local maximums

Solution results: My new algorithm was able to find a higher maximum energy. However, the probability that the quantum computer found this solution dramatically dropped. This challenges the assumption that increasing the Hamiltonian's energy will always increase the probability of the quantum computer finding the optimal solution.

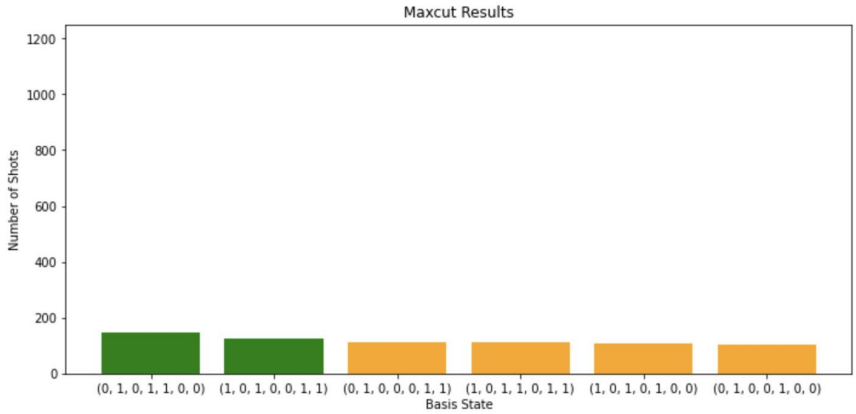
```
%%time
res = qaoa_calculate2(backend, backend.default_compilation_pass(1).apply, shots = 5000, iterations = 100, seed=12345)
```

```
highest energy:  5.208799999999999
best guess mixer angles:  [0.10873838 0.87274459 0.17330755]
best guess cost angles:  [0.21599029 0.74043714 0.45917754]
[(0, 1), (1, 2), (1, 3), (3, 4), (4, 5), (4, 6)]
CPU times: user 58.4 s, sys: 60.2 ms, total: 58.4 s
Wall time: 58.4 s
```

```
from maxcut_plotting import plot_maxcut_results

plot_maxcut_results(res, 6)
```

```
Success ratio 0.0544
```



The Hamiltonian just washed out over the basis states. I was able to alleviate this by creating a more complex initial state by adding X gates after the H gates.

```
%%time
res = qaoa_calculate2(backend, backend.default_compilation_pass(1).apply, shots = 5000, iterations = 100, seed=12345)
```

```
highest energy:  5.208799999999999
best guess mixer angles:  [0.10873838 0.87274459 0.17330755]
best guess cost angles:  [0.21599029 0.74043714 0.45917754]
[(0, 1), (1, 2), (1, 3), (3, 4), (4, 5), (4, 6)]
CPU times: user 57.9 s, sys: 83.9 ms, total: 57.9 s
Wall time: 57.9 s
```

```
from maxcut_plotting import plot_maxcut_results

plot_maxcut_results(res, 6)
```

```
Success ratio 0.1546
```