Quantum Algorithms for Max-cut Problems

Andris Huang

11 December, 2021

Abstract

The Maximum-cut (Max-cut) problem is an NP-complete problem studied by many researchers. Many real-life situations can be modeled as a Max-cut problem, and thus its solution is useful for many fields, including physics, chemistry, finance, etc. Given the recent development in quantum computing, there have been many approaches to find or approximate the solutions to Max-cut problems. This paper will explain two main quantum approaches for the Max-cut problem and examine their performances through a real-life example of a course scheduling system.

1 Introduction

The Max-cut problem is a type of graph problem that focuses on partition and grouping. A simple undirected graph can be defined as $G = (V, E)$, where V is the set of all vertices and E is the set of all edges. Each node x_i has an assigned value and each edge $e_{ij} \in E$ carries a numerical edge weight w_{ij} given by the context of a specific problem. The task for a Max-cut problem is to use one "cut" that disconnects an arbitrary number of edges and partitions the graph into two disjoint subsets S_0 and S_1, such that the sum of edges for the disconnected edges is maximized [1].

Mathematically, this problem can be formulated as

$$\max_x \left(C(x) = \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j) \right), \tag{1}$$

where $C(x)$ is the cost function and x_i is the value assigned to node i such that

$$x_i = \begin{cases} -1 & \text{if } i \in S_0 \\ 1 & \text{if } i \in S_1, \end{cases} \tag{2}$$

for the two subsets S_0 and S_1 after partition.

Due to the NP-hardness of this problem, it is difficult or impossible to find a classical algorithm that solves the Max-cut problem efficiently, since the complexity for the best general algorithm that solves problem classically is exponential in time $O\left(a^{N^{1/2}\log N}\right)$, where a is a constant [2]. As a result, most researchers spend their effort in finding an algorithm that can approximate the solution accurately. One of the best classical approximation algorithm has achieved a polylogarithmic time complexity of $O\left(n^2 \log n\right)$ with an approximation ratio of 0.8786 [3].

With the development of quantum mechanics, two major approaches to approximate the solutions to the Max-cut problems have become popular, namely, Quantum Annealing and the Quantum Approximate Optimization Algorithm. The next two sections of this paper will explain these two approaches in sequence.

2 Quantum Annealing

Quantum annealers are analog devices used to solve discrete combinatorial optimization problems using properties of quantum adiabatic evolution [2]. The foundation for quantum annealing is the Quantum Adiabatic Algorithm (QAA), proposed by Farhi et al. [4]. The fundamental theorem behind quantum annealing is that a physical system remains in its instantaneous eigenstate if the external perturbation acting on it is slow enough. As an extension, the state of the system is initialized in its instantaneous ground state, it also stays close to the instantaneous ground state of the Hamiltonian if its Hamiltonian varies slowly enough [1].

To conduct a quantum annealing process, two Hamiltonians are needed: an initial Hamiltonian H_B and a final Hamiltonian H_P. The overall Hamiltonian of the process is modeled by

$$\tilde{H}(s) = (1-s)H_B + sH_p, \tag{3}$$

where $s = t/T$ and T is the total annealing time [5], such that $s \in [0,1]$. In the initial stage when $s = 0$, the total Hamiltonian is equal to the initial Hamiltonian H_B and the final Hamiltonian vanishes. Similarly, when $s = 1$, the initial Hamiltonian vanishes and the total

Hamiltonian is equal to the final Hamiltonian.

Therefore, the process of quantum annealing is to first initialize a system in its ground state with Hamiltonian H_B, formulate another problem Hamiltonian H_P, and then slowly vary its total Hamiltonian by using external forces. According to the theorem described in the first paragraph of this section, the total Hamiltonian $\tilde{H}$ will remain in its instantaneous ground state as long as the variation is slow enough. When the system reaches $t = T$, the total Hamiltonian $\tilde{H}(s) = H_p$ will be in the ground state of and thus encode the lowest energy state solution to the problem [1].

For Max-cut problems, the system is typically initialized with

$$H_B = -\sum_i \sigma_i^x, \tag{4}$$

where σ^x is the Pauli x operator. The problem Hamiltonian is then formulated according to the cost function given by Eq. (1). Mathematically, it can be seen that $\max(f(x))$ is equivalent to $\min(-f(x))$. Therefore, the cost function in Eq. (1) is reformed into

$$\min_x \left(\tilde{C}(x) = \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(x_i x_j - 1) \right), \tag{5}$$

which then gives the Hamiltonian

$$H_p = \frac{1}{2} \sum_{(i,j)\in E} w_{ij} \left(\sigma_i^z \sigma_j^z - \mathbf{I} \right), \tag{6}$$

where σ^z is the Pauli z operator and E is the set of all edges in the graph [1][2]. The Pauli z operators are selected so that their eigenvalues $+1$ and -1 matches the possible values for x_i and x_j. After the annealing process, the total Hamiltonian will be equivalent to the problem Hamiltonian and be in its ground state, which gives the corresponding eigenvalues that minimizes the cost function in Eq. (5) and thus gives the solution to the problem.

3 Quantum Approximate Optimization Algorithm

Quantum Approximate Optimization Algorithm (QAOA) is a hybrid algorithm for suited for Noisy Intermediate Scale Quantum (NISQ) Devices [6]. The algorithm requires a quantum processor to prepare a quantum state according to a set of variational parameters that are optimized by a classical computer and fed back to the quantum machine in a closed loop. Similar to quantum annealing, major applications of QAOA include combinatorial optimization and quantum chemistry problems [5].

For the Max-cut problem, the state of the system is prepared by a p-level circuit specified by $2p$ variational parameters $(\vec{\gamma}, \vec{\beta}) = (\gamma_1, ..., \gamma_p, \beta_1, ..., \beta_p)$. It requires N qubits for a graph that contains N nodes, with each qubit representing a node as $|q_i\rangle$. If $|q_i\rangle$ is in the computional eigenbasis, then $\sigma_z|q_i\rangle = x_i|q_i\rangle$, where $x_i \in [-1, 1]$ is the node values defined in Eq. (2). The graph can thus be represented in a state as

$$|\psi\rangle = \otimes_{i=1}^{N} |q_i\rangle. \tag{7}$$

The problem Hamiltonian H_P needed is the same as the one used for quantum annealing, expressed by Eq. (6), such that

$$\begin{aligned} H_p|\psi\rangle &= \frac{1}{2} \sum_{(i,j)\in E} w_{ij} \left(\sigma_i^z \sigma_j^z |x_i x_j\rangle - \mathbf{I}|x_i x_j\rangle \right) \\ &= \left(\frac{1}{2} \sum_{(i,j)\in E} w_{ij} \left(x_i x_j - 1 \right) \right) |\psi\rangle, \end{aligned}$$

where the cost function to minimize in Eq. (5) is returned as the eigenvalue of the system when H_P is acted on it.

As a result, to solve a Max-cut problem, the QAOA circuit first initializes all the qubits in $|+\rangle$, and then alternatively applies $\mathbf{R}_x(\beta) = e^{-i\beta H_B}$ and $\mathbf{U}(\gamma) = e^{-i\gamma H_P}$ for P times, where H_B and H_P are defined in Eq. (4) and (6). The circuit for $\mathbf{R}_x(\beta)$ and $\mathbf{U}(\gamma)$ are shown in Figure 1. The final state is then measured to obtain an expectation value of the problem Hamiltonian as $\langle H_P \rangle$, which is then fed to a classical optimizer to find the best parameters $(\vec{\gamma}, \vec{\beta})$ that minimizes $\langle H_P \rangle$. The solution of the problem is the state of the system that achieves the minimal $\langle H_P \rangle$.

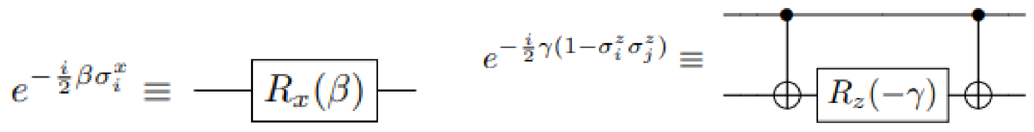
$e^{-\frac{i}{2}\beta\sigma_i^x} \equiv$
$R_x(\beta)$
$e^{-\frac{i}{2}\gamma(1-\sigma_i^z\sigma_j^z)} \equiv$
$R_z(-\gamma)$

Figure 1: The circuit diagrams for QAOA. The circuit on the left represents how to encode $e^{-i\beta H_B}$, which is just a rotation along the x direction for angle β. The circuit on the right encodes $\mathbf{U}(\gamma) = e^{-i\gamma H_P}$, which constructs **CNOT** gates between connected nodes i and j and applies a rotation in the z direction [7].

4 Application

4.1 Problem Formation

The real-life application studied in the paper is a course scheduler. The goal is to create a course scheduling system to allocate the time blocks for courses in a way that minimizes the time conflicts, so that the most number of students can attend the courses they desire with the least conflicts.

As a result, we encode the courses as the nodes of the graphs and constructs a fully-connected graph. We then assign the number of students taking the two courses concurrently as the weight of the edge that connects the corresponding two courses. After that, we perform the algorithm that solves the Max-cut problem (Max-cut solver) to separate the graph into two groups, then recursively apply it to the subgroups to separate them into multiple groups, until the number of courses in each subgroup is less than a threshold value given by the user. The implementation of this recursive algorithm is outlined in Algorithm 1.

Algorithm 1 Course Scheduling Algorithm: recursively applies the Max-cut solver until a threshold value is reached

```
$Graph$ = A fully-connected graph of all courses;
$threshold$ = Max number of courses per time block;        ▷ We used 10 as default
procedure SOLVER($G$)
    Separate input graph $G$ into $S_0$ and $S_1$ using the Max-cut solver;
    return $S_0$, $S_1$;
end procedure
procedure RECURSIVE-MAXCUT($G$)
    $V$ = Set of courses in $G$;
    $N$ = Length of $V$;
    if $N \leq threshold$ then
        return [V];
    else
        $S_0$, $S_1$ = SOLVER($G$);
        return RECURSIVE-MAXCUT($S_0$) + RECURSIVE-MAXCUT($S_1$);
    end if
end procedure
$Result$ = RECURSIVE-MAXCUT($Graph$).
```

The main motivations for studying this example is that firstly, this algorithm has the potential to effectively reduce the schedule conflicts at a large school like UC Berkeley. Secondly, it can be useful to study the run time of the algorithm with respect to the graph size through the recursive process, since the graph size is reduced in each recursive step.

4.2 Results

In this project, we randomly generated 150 courses from 6 departments, and assigned the number of students taking any two courses concurrently to be an integer in range $[1, 400]$. We first constructed a fully-connected graph with 150 nodes as shown in the left graph of Figure 2. We then perform the algorithm described in Algorithm 1 and separated the courses into 21 time blocks with less than 10 courses per block using quantum annealing, running on D-Wave's quantum annealer [8]. The resulting graph is shown on the right of Figure 2.

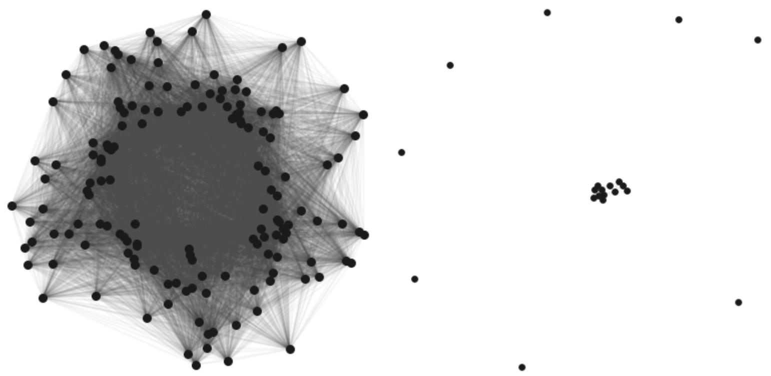

Figure 2: The left graph is the input fully-connected graph with 150 courses as nodes, number of scheduling conflicts as edge weights. The right graph is the resulting graph, with each cluster of nodes representing one time block. Each cluster contains less than 10 nodes.

We repeated this procedure on the same graph multiple times to collect more data points, and our best run reduced the total conflict by 95.58%, where the reduction proportion is calculated as

$$r = \frac{C(x_f)}{\sum_{(i,j)\in E} w_{ij}}, \tag{8}$$

where $C(x_f)$ is the total weights for disconnected edges and the denominator is the sum of edge weights before the partitions.

The run time with respect to graph size is shown in Figure 3. A reference line with $O(\exp(x))$ is also plotted. By comparison, it can be seen that the run time required for the model is still close to an exponential growth. Since the time required for an adiabatic quantum system to remain in the ground state is on $O\left(\exp(\alpha N^{\beta})\right)$ for a graph of size N and positive coefficients α and β [9], our result matches the theoretical bound.

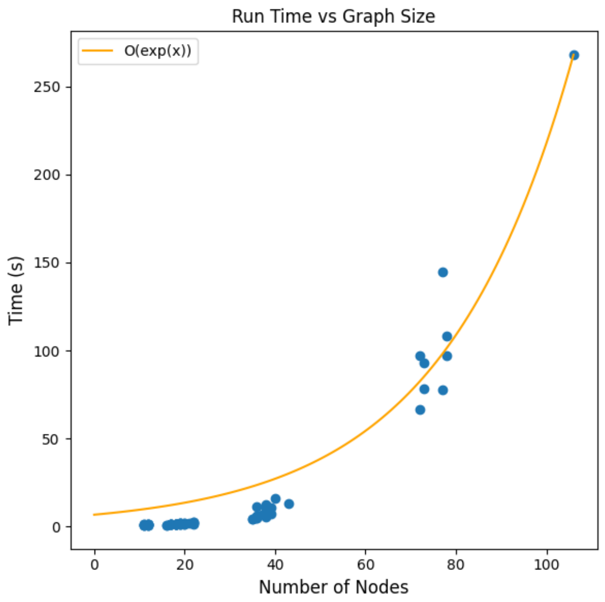

Figure 3: The run time of quantum annealing with respect to graph size.

We also tried to apply QAOA on the same graph. However, we were not able to obtain meaningful results since QAOA requires the same number of qubits as the number of nodes, indicating the requirement of 150 qubits, which is not achievable by any current system. Given the usable number of qubits for students, we were able to run QAOA with $p = 1$ on a random fully-connected graph with 20 nodes and a threshold value of 3 on the IBMQ Aer simulator to observe the run time, which is shown in Figure 4. Theoretically, the run time complexity for QAOA is $O(Np)$ for graph size N and circuit depth p [10]. However, it is worth noting that the actual run time also requires taking preparation time and measurement time into account, which can scale up with respect to the number of qubits. This could explain why the linear portion of our result only has a coefficient of determination r^2 value of 0.964, and the overall trend does not exhibit a linear behavior.

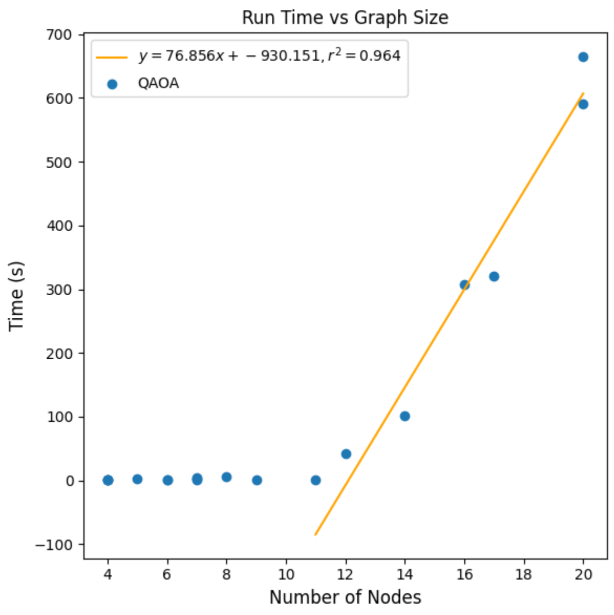

Figure 4: The run time of QAOA with respect to graph size. We ran the recursive algorithm multiple times to collect the data plotted.

As a comparison, the QAOA algorithm took 11.53 minutes to partition a 20-node graph into subgroups with threshold value of 3 and achieved a 93.02% reduction of conflicts, whereas quantum annealing took 7.98 seconds to partition the same group and achieved a 90.3% reduction of conflicts. However, this is not a conclusive comparison, since the Aer simulator cannot represent real quantum devices with more qubits, and we were only able to work with a limited number of qubits and a $p = 1$ depth algorithm.

5 Discussion

Although in theory quantum computing can provide significant speedup for approximating the solutions to Max-cut problems, the hardware has not been developed enough to supply the required qubits. Based on the comparison in the previous section, even though not conclusive, it can still be seen that quantum annealing is currently more effective to be put in practice, whereas QAOA is generally not usable for students or individual learners.

Future works on this project include using circuits of higher depth and more qubits to perform a comparable QAOA on our application. We could also implement the classical approach to observe whether quantum algorithms can provide a speedup as theorized.

In the future, it is also worth studying how quantum annealing and QAOA can be used to solve the Max k-cut problem [11] and whether using the Max k-cut algorithms can improve the reduction of conflicts and run time in our course scheduling example. It is also important to notice other applications that utilize the Max-cut model, including efforts to improve quarantine policies [12] and the incorporation of Max-cut into decision trees [13].

6 Conclusion

This paper studied two quantum approach to approximate the solutions to the Max-cut problems: quantum annealing and QAOA. We also implemented a recursive course scheduling algorithm using the Max-cut model as a real-life application to examine the two approaches in detail. Unfortunately, we observed that quantum annealing has an exponential growth in run time with respect to the graph size, whereas QAOA is not usable yet for this application due to limited accessible qubits. Nevertheless, both approaches enabled us to efficiently partition a graph with relatively small size, leaving potentials for future improvements.

7 Acknowledgement

This project is a group project with my two teammates Sidharth Duthaluru and Pranam Kalla. The application studied in this paper is inspired by other projects conducted in Quantum Computing Club at Berkeley. All codes used in this project can be found at https://github.com/Andris-Huang/qcb2021-22.

References

[1] F. McAndrew, "Adiabatic quantum computing to solve the maxcut graph problem," Ph.D. dissertation, The University of Melbourne, 2020.

[2] I. Hen and A. Young, "Solving the graph-isomorphism problem with a quantum annealer," *Physical Review A*, vol. 86, no. 4, p. 042 310, 2012.

[3] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[4] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[5] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, p. 021 067, 2020.

[6] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[7] G. E. Crooks, "Performance of the quantum approximate optimization algorithm on the maximum cut problem," *arXiv preprint arXiv:1811.08419*, 2018.

[8] H. Ushijima-Mwesigwa, C. F. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, 2017, pp. 22–29.

[9] A. Lucas, "Ising formulations of many np problems," *Frontiers in physics*, vol. 2, p. 5, 2014.

[10] G. G. Guerreschi and A. Y. Matsuura, "Qaoa for max-cut requires hundreds of qubits for quantum speed-up," *Scientific reports*, vol. 9, no. 1, pp. 1–7, 2019.

[11] F. G. Fuchs, H. Ø. Kolden, N. H. Aase, and G. Sartor, "Efficient encoding of the weighted max k-cut on a quantum computer using qaoa," *arXiv preprint arXiv:2009.01095*, 2020.

[12] J. Sud and V. Li, "A quantum annealing approach to reduce covid-19 spread on college campuses," *arXiv preprint arXiv:2112.01220*, 2021.

[13] J. Bodine and D. S. Hochbaum, "The max-cut decision tree: Improving on the accuracy and running time of decision trees," *arXiv preprint arXiv:2006.14118*, 2020.