NISQ algorithms 1: QAOA

Yipeng Huang

Rutgers University

September 23, 2020

Near-term intermediate-scale quantum (NISQ) computers

The limitations of near term quantum computers

- NISQ systems have limited number of qubits:
  No error correction.
  (In contrast, error corrected Shor's would need a million qubits.)
- NISQ systems have limited coherence time:
  Relative shallow depth of circuits.
  (In contrast, error corrected Shor's would need hundreds of millions of gates.)
- NISQ systems have limited operation accuracy

NISQ variational algorithms

Use a classical algorithm to train a "quantum neural network".

Figure: Credit: [Guerreschi and Smelyanskiy, 2017]

NISQ variational algorithms

Use a classical algorithm to train a "quantum neural network".

1. Quantum computer prepares a quantum state that is a function of classical parameters.
2. Quantum computer measures quantum state to provide classical observations.
3. Classical computer uses observations to calculate an objective function.
4. Classical computer uses optimization routine to propose new classical parameters to maximize objective function.
5. Repeating steps 1 through 4, the algorithm leads to better approximations to underlying problem.

NISQ variational algorithms

Bene ts of quantum-classical scheme:

1. Provides meaningful results even without error correction
2. Shallow circuits (not many operations on each qubit)
3. Draws on strengths of quantum and classical:
4. Prepare and measure a quantum state
5. Optimize for a set of optimal parameters based on classical measurements

NISQ variational algorithms

Great! Can NISQ variational algorithms solve useful problems?

1. Variational quantum eigensolver (VQE):
   Simulate quantum mechanics
2. Quantum approximate optimization algorithm (QAOA):
   Approximate solutions to constraint satisfaction problems (CSPs) [Farhi et al., 2014]

2.2 Combinatorial optimization problems

- Knapsack
- Traveling salesman
- Graph coloring
- MAX-SAT
- MAX-CUT

Boolean satis ability problem

- l-SAT: NP-Complete
- n bits; m constraints/clauses
- A Boolean formula consisting of m clauses
  $C_1 \wedge C_2 \wedge \cdots \wedge C_m$
  For example:
  $(\neg z_0 \vee z_1 \vee z_2) \wedge (\neg z_1 \vee z_2 \vee z_3) \wedge \cdots \wedge C_m$
- C depends only on l coordinates of z.
- Each clause C_α is either True or False.
  for each constraint $\alpha \in [m]$ and each n-bit string $z \in \{0,1\}^n$, de ne

$$C_\alpha(z) = \begin{cases} 1 & \text{if } z \text{ satis es the constraint} \\ 0 & \text{if } z \text{ does not} \end{cases}$$

Constraint satisfaction problem (CSP): MAX-SAT

- MAX-SAT: NP-Hard
- A Boolean formula consisting of m clauses
  $C_1 \wedge C_2 \wedge \cdots \wedge C_m$
- Each clause C is either True or False.
  for each constraint $\in [m]$ and each n-bit string $z \in \{0,1\}^n$, define
  $$C(z) = \begin{cases} 1 & \text{if } z \text{ satisfies the constraint} \\ 0 & \text{if } z \text{ does not} \end{cases}$$
- Satisfy as many clauses as possible to maximize objective function $C(z)$:
  $$\max_z C(z) = \max_z \sum_{=1}^{m} C(z)$$

Approximate MAX-SAT

Approximate the maximum:

$$\max_{\vec{z}} C(\vec{z}) = \max_{\vec{z}} \sum_{=1}^{m} C_{}(\vec{z})$$

Constraint satisfaction problem (CSP): MAX-CUT

- Given an arbitrary undirected graph $G = (V(G); E(G))$
- goal of MAX-CUT is to assign one of two partitions to each node so as to maximize the number of cuts

Figure: Credit: Quantum Algorithm Implementations for Beginners Coles.

Constraint satisfaction problem (CSP): MAX-CUT

- Given an arbitrary undirected graph $G = (V(G); E(G))$
- goal of MAX-CUT is to assign one of two partitions $_i \; 2 \; f \quad 1; +1 \, g$ to each node $i \; 2 \; V(G)$ so as to maximize the number of cuts
- Identical form to the MAX-SAT problem with objective function $C(\tilde{})$:

$$\max_{\tilde{}} C(\tilde{}) = \max_{\tilde{}} \sum_{< jk > 2 E(G)} C_{< jk >}(\tilde{})$$

- But the constraints are now:

$$C_{< jk >}(\tilde{}) = \frac{1}{2}(1 \quad _j \quad _k) = \begin{cases} 1 & \text{if } _j \text{ and } _k \text{ are di erent} \\ 0 & \text{if } _j \text{ and } _k \text{ are the same} \end{cases}$$

QAOA for MAX-CUT: general strategy

Figure: Credit: [Guerreschi and Smelyanskiy, 2017]

QAOA for MAX-CUT: general strategy

1. Each node in nodes of the MAX-CUT graph corresponds to one of n qubits in the quantum circuit.
2. The state vector across the qubits $|s\rangle$ encodes a node partitioning $\sim 2 \{-1, +1\}^n$
3. Put the initial state vector $|s\rangle$ in a superposition of all possible node partitionings
4. Need an operator (quantum gate) that encodes an edge $\langle jk \rangle \in E(G)$
5. Provide classical parameters such that the classical computer can control quantum partitioning
6. Perform a series of operations parameterized by classical parameters $\tilde{\gamma}$ and $\tilde{\beta}$ such that the final state vector $|(\tilde{\gamma}, \tilde{\beta})\rangle$ is a superposition of good partitionings
7. Optimize for a good set of $\tilde{\gamma}$ and $\tilde{\beta}$

QAOA for MAX-CUT: general strategy

Figure: Credit: Classical variational simulation of the Quantum Approximate Optimization Algorithm. Medvidovic and Carleo.

1. Each node in n nodes of the MAX-CUT graph corresponds to one of qubits in the quantum circuit.

Let's use an $n = 3$ example in the figure.

Figure: $G = (V(G); E(G)) = (\{q0; q1; q2\}; \{<q0q1>; <q1q2>\})$

$$|\ i = {}_0|000i + {}_1|001i ::: + {}_7|111i = \begin{matrix} 2 & 3 \\ & 0 \\ 6 & 7 \\ 6 & 1 \\ 4 & \ldots 5 \\ & \ldots \\ & 7 \end{matrix}$$

So now we have quantum amplitudes for each of the basis states.

Probability of measuring outcome z is $|\ _z|^2$. $\sum_{z=0}^{n-1} |\ _z|^2 = 1$

2. The state vector across the qubits $|i\rangle$ encodes a node partitioning

Figure: A max-cut corresponding to $|i\rangle = |010\rangle$.

- In our $n = 3$ running example, $|i\rangle = |010\rangle$, $\theta_2 = 1$, $\theta_0 = \theta_1 = \theta_3 = \theta_4 = \theta_5 = \theta_6 = \theta_7 = 0$, is a max-cut.
- The other max-cut is $|i\rangle = |101\rangle$.
- A superposition $|i\rangle = \frac{1}{\sqrt{2}}|010\rangle + \frac{1}{\sqrt{2}}|101\rangle$ would be an equal superposition of the two max cuts.

3. Put the initial state vector $|\psi_s\rangle$ in a superposition of all possible node partitionings

Figure: Credit: How many qubits are needed for quantum computational supremacy. Dalzell et al.

3. Put the initial state vector $|\psi_s\rangle$ in a superposition of all possible node partitionings

A superposition across all the bitstrings representing partitionings.

$$|\psi_s\rangle = |+\rangle^{\otimes n} = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} |z\rangle$$

3. Put the initial state vector $|\psi_s\rangle$ in a superposition of all possible node partitionings

In our $n = 3$ running example:

$$|+\rangle^{\otimes n} = H^{\otimes 3}|0\rangle^{\otimes 3} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{\otimes 3} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^{\otimes 3} =$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{\otimes 3} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^{\otimes 3} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{\otimes 3} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^{\otimes 3} =$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{\otimes 3} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^{\otimes 3} =$$

$$\frac{1}{\sqrt{8}} \sum_{z=0}^{7} |z\rangle = [\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}]^{T}$$

4. Need an operator (quantum gate) that encodes an e

$< jk > 2 E(G)$

- In the Max-Cut type of CSP, constraints correspond to edges
- $C = \sum_{<jk>2E(G)} C_{<jk>} = \sum_{<jk>2E(G)} \frac{1}{2}(1 \quad {}^{z}_{j} \quad {}^{z}_{k})$
- ${}^{z}_{i}$ is the Pauli-Z operator on qubit i
- ${}^{z} = \begin{matrix} +1 & 0 \\ 0 & 1 \end{matrix}$
- Claim: that maximizesh $j C j$ i is the graph partition with the maximum cut.

4. Need an operator (quantum gate) that encodes an e

$< jk > 2\ E(G)$

In our n = 3 running example:

Figure: $G = (V(G); E(G)) = (\{q0; q1; q2g; \{< q0q1 >; < q1q2 >\})$

$$C = \sum_{<jk>2E(G)} \frac{1}{2}(1 - \sigma_j^z \sigma_k^z) = \frac{1}{2}(1 - \sigma_0^z \sigma_1^z) + \frac{1}{2}(1 - \sigma_1^z \sigma_2^z) =$$

$$\frac{1}{2}(I \otimes I - \sigma^z \otimes \sigma^z) + \frac{1}{2}(I \otimes I - \sigma^z \otimes \sigma^z) =$$

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

4. Need an operator (quantum gate) that encodes an e
< jk > 2 E(G)

Figure: A max-cut corresponding to $|i\rangle = |0101\rangle$.

- Claim: $|j\rangle$ that maximizes $\langle j|C|j\rangle$ is the graph partition with the maximum cut.
- $\langle 0101|C|0101\rangle =$ $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} = 2$

4. Need an operator (quantum gate) that encodes an e
$< jk > 2 E(G)$

- $\langle 010 | C | 010 \rangle = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} = 2$
- Note the size of state vector $|i\rangle$ is 2^n.
- Size of constraint matrix C is $2^n \times 2^n$.
- We will enlist a quantum computer to create $|i\rangle$ and C.

5. Provide classical parameters such that the classical computer can control quantum partitioning

Figure: Credit: How many qubits are needed for quantum computational supremacy. Dalzell et al.

5. Provide classical parameters such that the classical computer can control quantum partitioning

- A parameterp that controls how many iterations of algorithm and how complete of a graph to see.
- Do optimization across p 2-dimensional vector of ~ and ~ parameters
- (~; ~) = ($_1$; $_1$; ::: $_p$; $_p$)
- $_i$ 2 [0; 2]
- $_i$ 2 [0;]

6. Perform a series of operations parameterized by clas

parameters~ and ~ such that the nal state vector
j (~; ~)i is a superposition of good partitionings

Figure: Credit: How many qubits are needed for quantum computational supremacy. Dalzell et al.

6. Perform a series of operations parameterized by clas

parameters $\tilde{\gamma}$ and $\tilde{\beta}$ such that the final state vector $|\tilde{\gamma}, \tilde{\beta}\rangle$ is a superposition of good partitionings

$U(C, \gamma)$, $U(B, \beta)$ are $2^n \times 2^n$ linear operators (Unitary matrices)

1. Problem Hamiltonian enforces constraints.
   A product across all the graph edges.
   $$U(C, \gamma) = e^{-i\gamma C} = \prod_{<jk> \in E(G)} e^{-i\gamma C_{<jk>}}$$

2. Admixing Hamiltonian perturbs the assignments.
   A product across all the qubits representing graph vertices.
   $$U(B, \beta) = e^{-i\beta B} = \prod_{q \in V(G)} e^{-i\beta \sigma_q^x}$$

6. Perform a series of operations parameterized by clas

parameters~ and ~ such that the nal state vector
j (~; ~)i is a superposition of good partitionings

Figure: Credit: How many qubits are needed for quantum computational supremacy. Dalzell et al.

6. Perform a series of operations parameterized by clas

parameters $\tilde{\gamma}$ and $\tilde{\beta}$ such that the final state vector $|\psi(\tilde{\gamma},\tilde{\beta})\rangle$ is a superposition of good partitionings

- Create ansatz state $|\psi(\tilde{\gamma},\tilde{\beta})\rangle$
- $|\psi(\tilde{\gamma},\tilde{\beta})\rangle = U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_1)U(C,\gamma_1)|s\rangle =$

$$\prod_{i=1}^{p} \prod_{q\in V(G)} e^{-i\beta_i X_q} \prod_{<jk>\in E(G)} e^{-i\gamma_i C_{<jk>}} |+\rangle^n$$

7. Optimize for a good set of and ~

- Measure this state to compute the objective function. That is, given the current set of parameters and ~, how much of the CSP is satisi ed
- Use a classical optimization algorithm such as Nelder-Mead to maximize $F_p(\sim; \sim) = h\ (\sim; \sim)j\, C\, j\ \ (\sim; \sim)i$

Evaluation of QAOA for NISQ: Number of iterations?

- Let M_p be the maximum of F_p over the angles:
  $M_p = \max_{\tilde{\gamma}, \tilde{\beta}} F_p(\tilde{\gamma}, \tilde{\beta})$
- QAOA needs a big parameter p to see the whole graph
- As $p \to \infty$, $\lim_{p \to \infty} M_p = \max_z C(z)$

Evaluation of QAOA for NISQ: Number of iterations?

Figure: Credit: [Arute et al., 2020]

Evaluation of QAOA for NISQ: Number of qubits?

Figure: Credit: [Guerreschi and Matsuura, 2019]

Evaluation of QAOA for NISQ: Number of constraints?

- Findings for MAX-CUT on connected 3-regular graphs [Farhi et al., 2014]
  1. For p = 1, QAOA will always produce a cut whose size is at least 0.6924 times the size of the optimal cut.
  2. This was the best known possible approximation for a few months until classical algorithm found.
  3. For p = 2, the approximation ratio becomes 0.7559 and grows depending on the type of graph.
- Di culty of solving problems with higher constraint ratios [Akshay et al., 2020]

Evaluation of QAOA for NISQ: Optimization method?

Optimization using gradients [Guerreschi and Smelyanskiy, 2017].

Evaluation of QAOA for NISQ: Generalizations?

- Can be generalized to solve related problems [Had eld et al., 2019].
- Factoring [Anschuetz et al., 2019].

Read also

- Primary sources: [Farhi et al., 2014]
- Additional source: [Wang and Abdullah, 2018]

Akshay, V., Philathong, H., Morales, M. E. S., and Biamonte, J. D. (2020).
Reachability de cits in quantum approximate optimization.
Phys. Rev. Lett., 124:090504.

Anschuetz, E., Olson, J., Aspuru-Guzik, A., and Cao, Y. (2019).
Variational quantum factoring.
In Feld, S. and Linnho -Popien, C., editors, Quantum Technology and Optimization Problems, pages 74{85, Cham. Springer International Publishing.

Arute, F. C., Arya, K., Babbush, R., Bacon, D., Bardin, J., Barends, R., Boixo, S., Broughton, M. B., Buckley, B. B., Buell, D. A., Burkett, B., Bushnell, N., Chen, J., Chen, Y., Chiaro, B., Collins, R., Courtney, W., Demura, S., Dunsworth, A., Farhi, E., Fowler, A., Foxen, B. R., Gidney, C. M., Giustina, M., Gra , R., Habegger, S., Harrigan, M., Hong, S., Io e, L., Isakov, S., Je rey, E., Jiang, Z., Jones, C., Kafri, D., Kechedzhi, K., Kelly, J., Kim, S., Klimov, P., Korotkov, A., Kostritsa, F., Landhuis, D., Laptev, P., Leib, M., Lindmark, M., Lucero, E., Martin, O., Martinis, J., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Mohseni, M., Mruczkiewicz, W., Mutus, J., Naaman, O., Neeley, M., Neill, C., Neukart, F., Neven, H., Niu, M. Y., O'Brien, T. E., O'Gorman, B., Petukhov, A., Putterman, H., Quintana, C., Roushan, P., Rubin, N., Sank, D., Satzinger, K., Skolik, A., Smelyanskiy, V., Strain, D., Streif, M., Sung, K. J., Szalay, M., Vainsencher, A., White, T., Yao, J., Zalcman, A., and Zhou, L. (2020).
Quantum approximate optimization of non-planar graph problems on a planar superconducting processor.
arXiv:2004.04197.

Farhi, E., Goldstone, J., and Gutmann, S. (2014).
A quantum approximate optimization algorithm.

arXiv preprint arXiv:1411.4028.

Guerreschi, G. G. and Matsuura, A. Y. (2019).
Qaoa for max-cut requires hundreds of qubits for quantum speed-up.
Scienti c Reports, 9(1):6903.

Guerreschi, G. G. and Smelyanskiy, M. (2017).
Practical optimization for hybrid quantum-classical algorithms.
arXiv preprint arXiv:1701.01450.

Hadfield, S., Wang, Z., O'Gorman, B., Rieffel, E. G., Venturelli, D., and Biswas, R. (2019).
From the quantum approximate optimization algorithm to a quantum alternating operator ansatz.
Algorithms, 12(2):34.

Wang, Q. and Abdullah, T. (2018).
An introduction to quantum optimization approximation algorithm.