Overparameterization in the quantum approximate optimization algorithm

Rebekah Herrman

University of Tennessee

rherrma2@utk.edu

Thanks

Jim Ostrowski (UTK), Phil Lotshaw (ORNL), Travis Humble (ORNL), and George Siopsis (UTK)

Kaiyan Shi (UMD), Ruslan Shaydulin (JPMC), Shouvanik Chakraborty (JPMC), Marco Pistoia (JPMC), Jeff Larson (ANL)

Outline of Presentation

- Motivation
- QAOA Background
- ma-QAOA
  - Theoretical results
  - Determining parameters
- Symmetry and ma-QAOA

Why quantum?

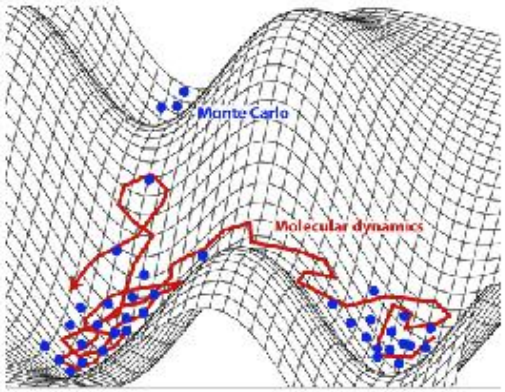


Figure: Image from Wikipedia

Figure: Image from Wikipedia

QAOA Background

- Quantum approximate optimization algorithm (QAOA)
- Introduced in 2014 by Farhi, Goldstone, and Gutmann to approximately solve combinatorial optimization problems [3]
  - MaxCut, MaxIndSet are two commonly studied ones
- Algorithm has gained a lot of attention in recent years [4, 2, 7]
  - Promising experimental results
  - Lots of open problems

What is a combinatorial optimization problem?

- Specified by n bits and m clauses.
  - Clauses are constraints on a subset of bits
- Define an objective function, which is the number of clauses satisfied:

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

  where $z = x_1 ... x_n$ is a bit string.
  - $C_\alpha(z) = 1$ if clause α is satisfied, else it is 0
- Is there a bit string that satisfies all of the clauses? How can we maximize or minimize $C(z)$?

Example: MaxCut

- Let $G = (V, E)$ be a graph. We want to divide V into two sets such that the number of edges between them is maximized.
- This problem can be formulated as

$$\min_{x \in \{0,1\}^n} \sum_{ij \in E(G)} x_j(x_i - 1) + x_i(x_j - 1) = \min_{x \in \{0,1\}^n} \sum_{ij \in E(G)} 2x_i x_j - x_i - x_j$$

Example: MaxCut

We will call the two sets red and blue.

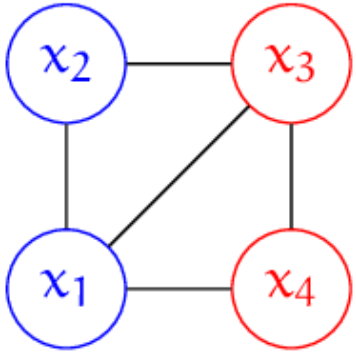


Figure: Not the best MaxCut solution. There are three edges whose endpoints are in different sets.

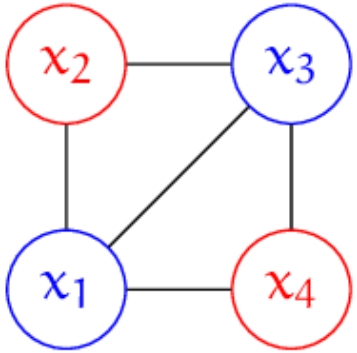


Figure: The best MaxCut solution. There are four edges whose endpoints are in different sets.

How are these problems solved with QAOA?

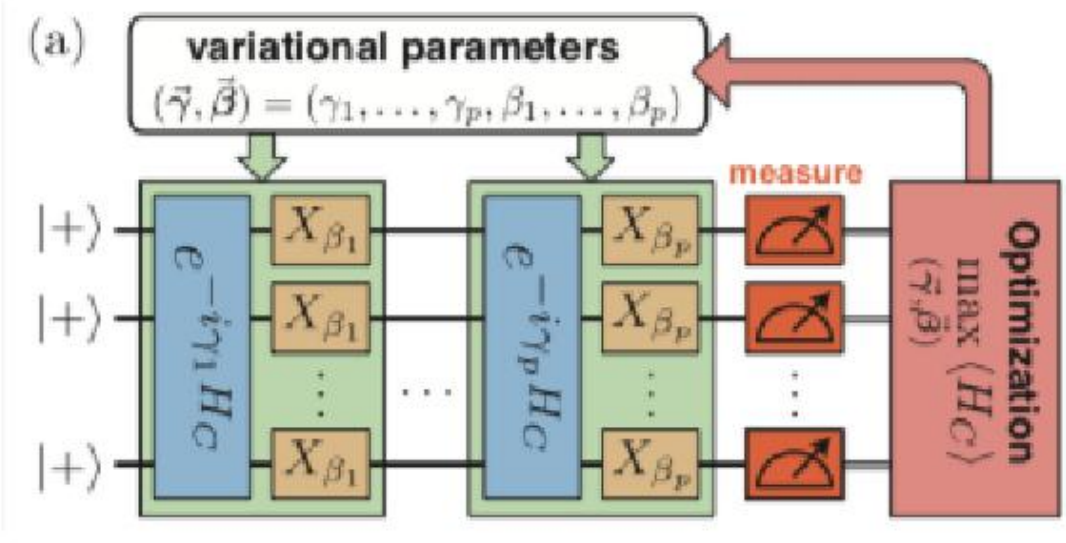


Figure: QAOA [1]

How are these problems solved with QAOA?

- We want to create a unitary operator based on C:

$$U(C,\gamma) = e^{-iC\gamma} = \prod_{\alpha} e^{-iC_\alpha \gamma}$$

- Alternate this operator with a mixing operator

$$U(B,\beta) = e^{-iB\beta}$$

  and apply to an initial state $|s\rangle$ that depends on B.
- In the end, get $|\gamma,\beta\rangle = U(B,\beta_p)U(C,\gamma_p)...U(B,\beta_1)U(C,\gamma_1)|s\rangle$ which has depth p. γ and β are angles.
- The approximation ratio is $\langle C\rangle / C_{max}$ and is primary metric for success.

Motivation

- There is a trade off between time spent using classical techniques and quantum techniques when implementing QAOA
- By shifting more burden onto classical optimization, we can design shallower quantum circuits that might be implementable in the NISQ era

Multi-angle QAOA

- How much better can we do if we allow each gate to have its own angle?
- We redefine C and B so that each summand C_a and B_v has its own angle, γ_a and β_v respectively to get

$$U(C, \vec{\gamma}) = e^{-iC\vec{\gamma}} = e^{-i\sum_a C_a \gamma_a} = \prod_a e^{-i\gamma_a C_a}$$

and

$$U(B, \vec{\beta}) = e^{-iB\vec{\beta}} = e^{-i\sum_{v \in V(G)} B_v \beta_v} = \prod_{v \in V(G)} e^{-i\beta_v B_v}$$

where $\vec{\gamma} = (\gamma_{a_1}, \gamma_{a_2}, ...)$ and $\vec{\beta} = (\beta_{v_1}, \beta_{v_2}, ...)$.

Multi-angle QAOA

Theorem

[5] The multi-angle quantum approximate optimization algorithm converges to the optimal solution as $\mathrm{p} \to \infty$.

- Prove using squeeze theorem
- The theorem ignores the difficulty of optimizing angle selection.
- We don't need exact optimal, we just need to do better than the original QAOA.

Multi-angle QAOA

Theorem

[5]
Let $\beta'_u = 2\beta_u$ *and* $\beta'_v = 2\beta_v$ *The expected value of* C *after one iteration of ma-QAOA applied to MaxCut for triangle free graphs* G *is* $\sum_{uv\in E} \langle \gamma_{uv}\beta_x | C_{uv} | \gamma_{uv}\beta_x \rangle$, *where:*

$$\langle \gamma_{uv}\beta_x | C_{uv} | \gamma_{uv}\beta_x \rangle = \frac{1}{2} + \frac{1}{2}\sin\gamma_{uv}(\cos\beta'_v \sin\beta'_u \prod_w \cos\gamma_{uw} + \cos\beta'_u \sin\beta'_v \prod_x \cos\gamma_{vx})$$

where $w \in \mathrm{Nbhd}(u) \setminus v$ *and* $x \in \mathrm{Nbhd}(v) \setminus u$.

- The proof relies on the Pauli-solver algorithm
- This looks a lot like the equations for weighted graphs.

Preliminary Data

- We compare one iteration of ma-QAOA to 1-QAOA, 2-QAOA, and 3-QAOA for solving MaxCut on all non-isomorphic, connected eight-vertex graphs.

QAOA type	Average approximation ratio for all eight-vertex graphs
ma-QAOA	.9257
p=1 QAOA	.8061
p=2 QAOA	.8767
p= 3 QAOA	.9192

Table: The average approximation ratio for eight-vertex graphs.

- Used BFGS to find optimal angles
- Over 20% of graphs have an approximation ratio of 1.0 for one iteration of ma-QAOA.

Theoretical Result

- ma-QAOA has an approximation ratio of 1 for MaxCut on all star graphs
- QAOA approximation ratio tends to 0.75 for large n

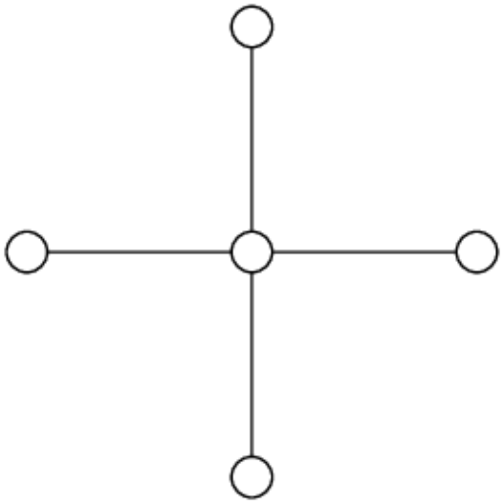


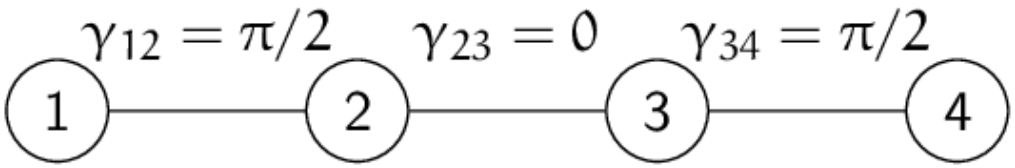
Figure: The star graph on five vertices.

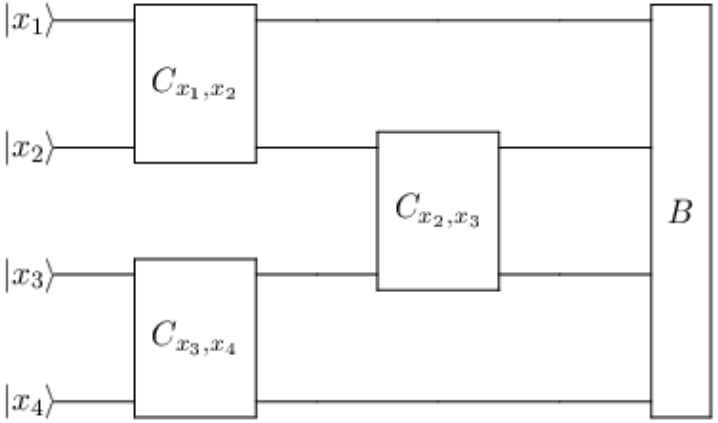
Efficient Embedding

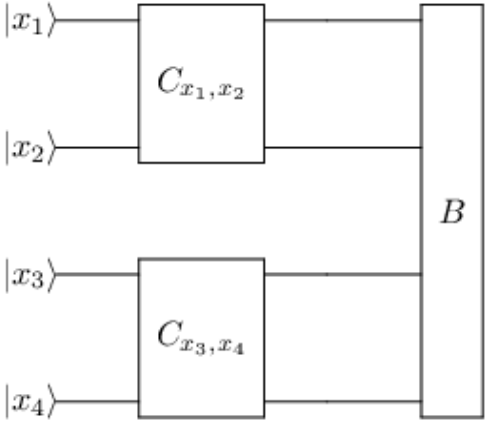
- The above results indicate that ma-QAOA may need fewer iterations than QAOA. This alone should lead to shallower circuits.
  - 15% of β angles are zero.
  - 25% of γ angles are zero.
- Zero-angle gates don't have to be implemented, leading to smaller circuits (especially on limited-connection hardware- fewer swaps).

Example: Multi-angle QAOA and circuit depth

- For example, consider P_4 .







Drawback

- Lots of angles are needed for this: n beta angles and up to $\binom{n}{2}$ gamma angles per iteration

Graph Automorphisms

- A *graph automorphism* is a permutation of the vertex set of a graph, $\sigma : V \to V$
  - Satisfies the condition that a pair of vertices, (u, v), forms an edge if and only if the pair $(\sigma(u), \sigma(v))$ also forms an edge.
- Any set of automorphisms generate a corresponding vertex and edge orbits, which are the equivalence classes of the vertices (edges) under the action of the automorphism.
- A *generator* of a group is a set of automorphisms $(\sigma_1, \ldots, \sigma_n)$ containing group elements so that application of the generators on themselves and each other can produce all elements in the group.

sym-QAOA

- **Idea:** use problem symmetry to reduce the number of parameters in ma-QAOA [6]
- sym-QAOA selects a single automorphism and assigns the same angle to all vertices in the same vertex orbit and the same angle to all edges in the same edge orbit
- best-1sym-qaoa runs sym-QAOA over all automorphisms and selects the one that gives the largest Max-Cut value
- Requires $|\mathrm{Orb}_v| + |\mathrm{Orb}_e|$ parameters
  - each element in the same vertex orbit or edge orbit receives the same parameter

max-sym-QAOA and rand-group-QAOA

- max-sym-QAOA computes the symmetry generator of G and determine the corresponding maximum vertex orbit, Orb_v, and edge orbit, Orb_e.
- Finding the generating set of the automorphism group of a graph is an extra step in max-sym-QAOA
- rand-group-qaoa groups vertices in the problem graph randomly into sets and edges randomly into sets, such that the number of parameters is the same as that of max-sym-qaoa.

best-1sym-QAOA vs. ma-QAOA

- Below figure shows the difference in approximation ratios between best-1sym-qaoa and ma-qaoa for these $5,918$ graphs.
- best-1sym-qaoa has the same approximation ratio as ma-qaoa has on $5,097$ graphs ($\sim 86.1\%$ of the studied graphs)
- On average used 28.1% fewer parameters

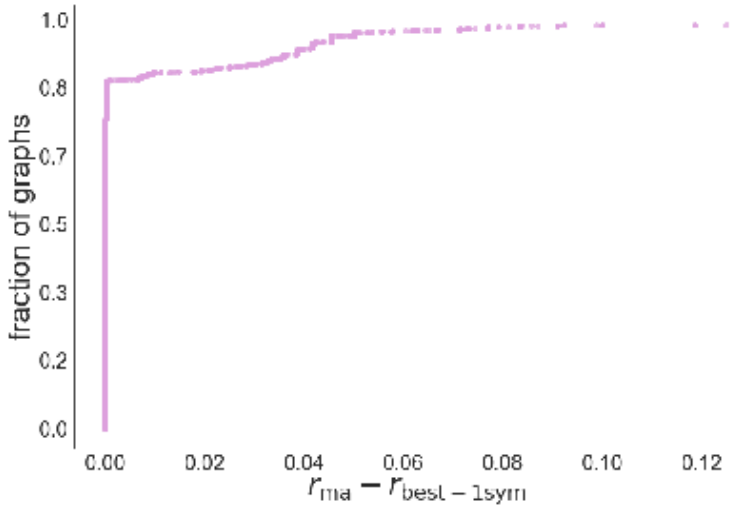


Figure: Difference in approximation ratios.

max-sym-QAOA vs. ma-QAOA

- Again, compare max-sym-QAOA to ma-QAOA on the $7,565$ graphs that contain nontrivial symmetry.
- As shown in the below figure, max-sym-QAOA performs as well as ma-QAOA on 2,713 of these graphs.

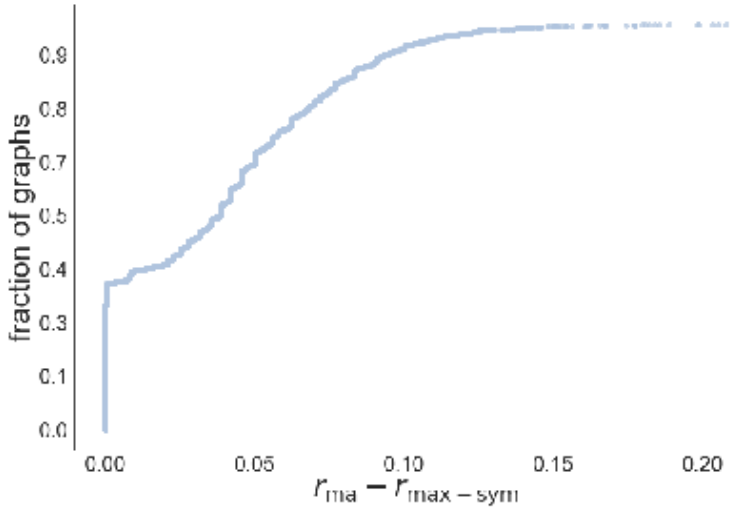


Figure: Difference in approximation ratios.

Evidence for central role of symmetries

- Compare max-sym-QAOA to rand-group-QAOA on 7,565 graphs where max-sym-QAOA does not equal ma-QAOA
- Define the ratio

$$k_{\text{var}} := \frac{f(x^*_{\text{ma}}) - f(x^*_{\text{var}})}{f(x^*_{\text{ma}}) - f(x^*)}.$$

- $k = 0$, the QAOA variant recovers ma-QAOA, and when $k = 1$, it performs the same as QAOA.

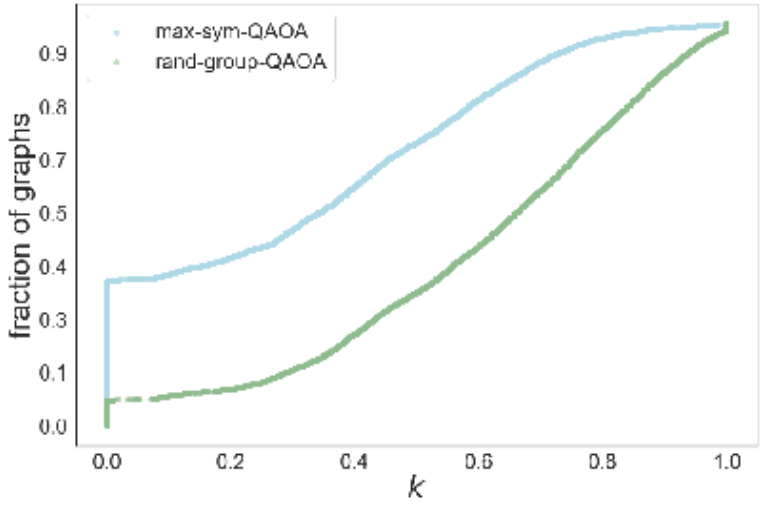


Figure: Fraction of graphs achieving ratio k.

Current work/ future directions

- Mathematically prove ma-QAOA = best-1sym-QAOA for particular kinds of graphs
- ma-QAOA performance bounds
- Are there other classes of graphs where ma-QAOA strictly outperforms QAOA?

References

[1] Found at https://medium.com/mit-6-s089-intro-to-quantum-computing/qaoa-bench-marking-7dfdd8a31e54.

[2] E. Farhi, D. Gamarnik, and S. Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples. *arXiv preprint arXiv:2005.08747*, 2020.

[3] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[4] G. G. Guerreschi and A. Y. Matsuura. Qaoa for max-cut requires hundreds of qubits for quantum speed-up. *Scientific reports*, 9, 2019.

[5] R. Herrman, P. C. Lotshaw, J. Ostrowski, T. S. Humble, and G. Siopsis. Multi-angle quantum approximate optimization algorithm. *arXiv preprint arXiv:2109.11455*, 2021.

[6] K. Shi, R. Herrman, R. Shaydulin, S. Chakrabarti, M. Pistoia, and J. Larson. Multi-angle qaoa does not always need all its angles. *arXiv preprint arXiv:2209.11839*, 2022.

[7] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97(2):022304, 2018.

Thanks