Quantum Approximate Optimization Algorithms

Smit Chaudhary, Ignacio Fernández Graña and Luigi Pio Mastrodomenico

Delft University of Technology, The Netherlands

January 14, 2021

Outline

Outline

Combinatorial optimization problems

- **Goal** Minimize/maximize a given objective function $C(x)$

$$\begin{aligned} \text{maximize} \quad & C(x) \\ \text{subject to} \quad & x \in S \end{aligned}$$

- **Problem**: Very fast growing discrete solution space $\rightarrow$ NP Hard problems

Max-Cut problem

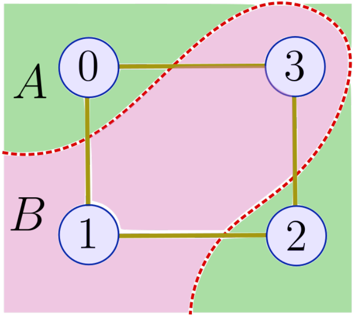
Given a graph $G(V, E)$, partition the set of vertices V into two sets A and B such that the number of edges being cut is maximised.

The Max-Cut problem

- Graph partition represent by a **bitstring** of length n (e.g. $x = ABAB \equiv 0101$)
- **Goal**: maximize objective function

$$C_{jk} = \frac{1}{2}(1 - (-1)^{x_j + x_k})$$

$$C(x) = \sum_{edges(jk)}^{m} C_{jk},$$



Max-Cut partition for a 4-n graph[1]

[1] https://pennylane.ai/qml/demos/tutorial_qaoa_maxcut.html

QAOA for the Max-Cut

Quantum Approximate Optimization Algorithm (QAOA): A hybrid quantum-classical approximate algorithm

- Map graph partitions to states in the Hilbert space: $x = |0101\rangle$
- Encode the cost function into a Hamiltonian

$$H_C = \sum_{<jk>} \frac{1}{2}(1 - Z_j Z_k) = \sum_{x \in \{0,1\}^n} C(x)|x\rangle\langle x|$$

- H_B: Mixing Hamiltonian.

$$H_B = \sum_j^n \sigma_j^x = \sigma_1^x \otimes \mathbb{I}^{\otimes(n-1)} + \mathbb{I}_1 \otimes \sigma_2^x \otimes \mathbb{I}^{\otimes(n-2)} + \cdots + \mathbb{I}^{n-1} \otimes \sigma_n^x$$

- Apply a set of operators to the maximal superposition state $|+\rangle^{\otimes n}$

$$U_C(\gamma_i) = e^{-i\gamma_i H_C} \qquad U_B(\beta_i) = e^{-i\beta_i H_B}$$

$$|\gamma, \beta\rangle = U_B(\beta_p) U_C(\gamma p) \ldots U_B(\beta_1) U_C(\gamma_1)|+\rangle^{\otimes n}$$

QAOA in Action

$$\gamma = 0, \beta = 0$$

$$U_B U_C |\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix} \implies \mathrm{C} = 0.5$$

QAOA in Action

$$\gamma = \pi, \beta = 0.1124$$

$$U_B U_C |\psi\rangle = U_B \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.487 + 0.112i \\ -0.487 - 0.112i \\ -0.487 - 0.112i \\ 0.487 + 0.112i \end{bmatrix}$$

$$P = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix} \qquad \implies \mathrm{C} = 0.5$$

QAOA in Action

$$\gamma = 1.0046, \beta = 1.1278$$

$$U_B U_C |\psi\rangle = \begin{bmatrix} 0.011 - 0.208i \\ -0.17 - 0.654i \\ -0.17 - 0.654i \\ 0.011 - 0.208i \end{bmatrix}$$

$$P = \begin{bmatrix} 0.043 \\ 0.457 \\ 0.457 \\ 0.043 \end{bmatrix} \implies \mathrm{C} = 0.913$$

QAOA in Action

$$\gamma = -1.5708, \beta = 0.3927$$

$$U_B U_C |\psi\rangle = \begin{bmatrix} 0 \\ -0.707i \\ -0.707j \\ 0 \end{bmatrix}$$

$$P = \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \\ 0 \end{bmatrix} \qquad \Longrightarrow \quad \mathrm{C} = 1$$

QAOA in Action

$$\gamma = -0.6663, \beta = 0.5$$

$$U_B U_C|\psi\rangle = \begin{bmatrix} 0.010 - 0.331i \\ 0.212 - 0.588i \\ 0.212 - 0.588i \\ 0.010 - 0.331i \end{bmatrix}$$

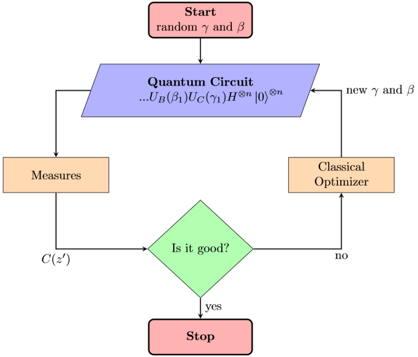
$$P = \begin{bmatrix} 0.109 \\ 0.391 \\ 0.391 \\ 0.109 \end{bmatrix} \implies \mathrm{C} = 0.781$$

QAOA

- $\gamma, \beta = \gamma_p, \beta_p \dots \gamma_1, \beta_1$ optimized via **classical optimization** of the *average value of the cost function* $F(\gamma, \beta)$

$$F(\gamma, \beta) = \langle \gamma, \beta | \, H \, | \gamma, \beta \rangle$$

- Final state $|\gamma, \beta\rangle$ is such that the solution state has a large probability of being measured



Outline

Implementation

Building U_C and U_B

- $U_C(\gamma) = \prod_{i,j}^{edges} CR_{Z\,i,j}(-2\gamma) R_{Zi}(\gamma) R_{Zj}(\gamma)$
  - $CR_Z(-2\gamma)$: controlled rotation around z with angle -2γ

$$CR_Z(-2\gamma) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i2\gamma} \end{pmatrix}$$

  - $Rz_i(\gamma)$: Z-rotation with angle γ:
- $U_B(\beta) = \prod_{i=1}^{n} R_{Xi}(2\beta)$
  - Rx: X-rotation of all the qubits with angle 2β

Implementation: Quantum Circuit scheme

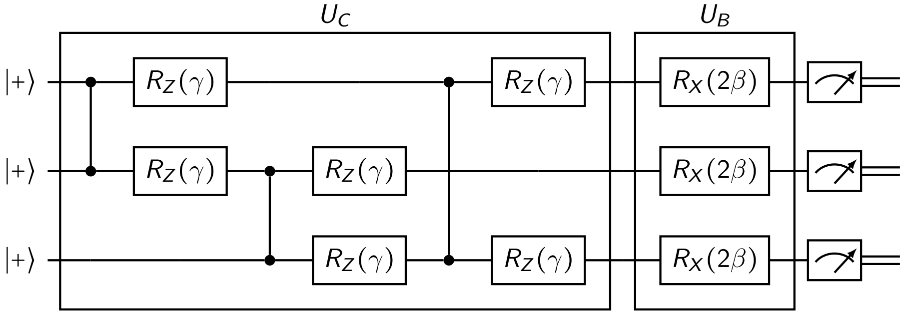
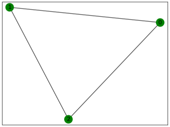


Figure: Circuit for p=1 on the 3-n regular graph with the final measurement in the computational basis



Implementation

- **Qiskit**
- Global **optimizer**: Differential Evolution[2]

$$b' = b_0 + mutation * (population[rand0] - population[rand1])$$

- **Metric** of performance: approximation ratio r

$$r = \frac{C_{average}}{C_{max}}$$

[2] Storn, R and Price, K, Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 1997, 11, 341 - 359.

State vector simulator

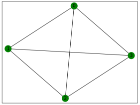


Figure: 4-n regular graph

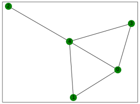


Figure: Erdos-Renyi 5 nodes

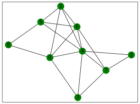
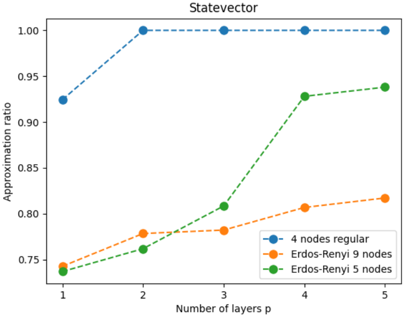
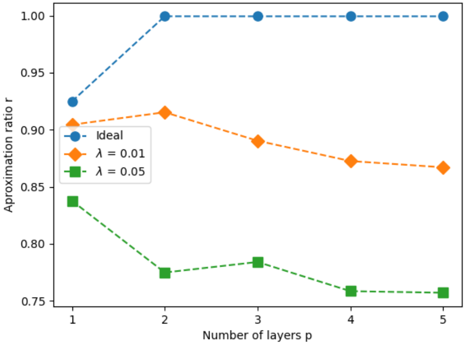


Figure: Erdos-Renyi 9 nodes



- Statevector simulator: most ideal case. Nothing quantum, just algebra!
  – Not possible to implement noise
- From now on: 4-n regular graph

QASM Simulator



- With QASM we can model noise (e.g. depolarizing channel)

$$\rho \to (1-\lambda)\rho + \lambda \frac{I}{d}$$

- Errors add up for $p > 1$

Towards real backend: FakeVigo

Fake Vigo simulator from Qiskit Aer

- Single and 2-qubits gate **depolarizing** error
- T1, T2 **relaxation times** of each qubit.
- **Readout** error

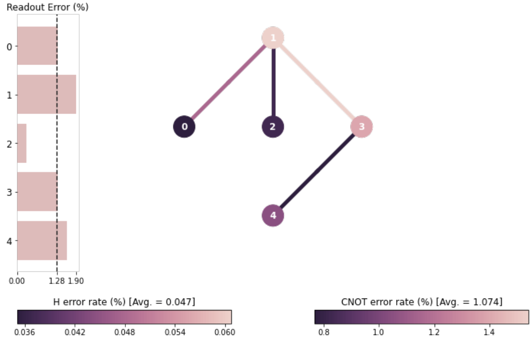
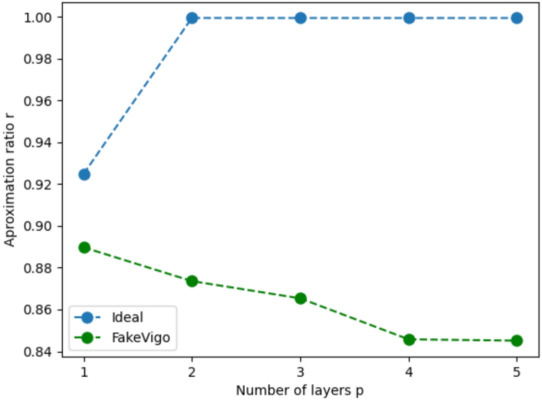


Figure: Fake Vigo error map[3]

[3] $ibmq_V igo$ v1.0.2, IBM Quantum team. Retrieved from https://quantum-computing.ibm.com (2020).

QAOA performance in FakeVigo



- **Simplified** noise model
- Optimal p is 1

Hardware backend

IBM Quantum Experience: Vigo

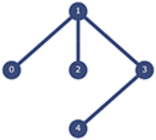


Figure: IBM Vigo qubits[4]

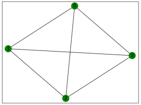


Figure: 4-n regular graph

- Access via **cloud**
- Every **evaluation** of the objective function is expensive

[4] *ibmq$_V$igo* v1.0.2, IBM Quantum team. Retrieved from https://quantum-computing.ibm.com (2020).

Layer by layer optimization

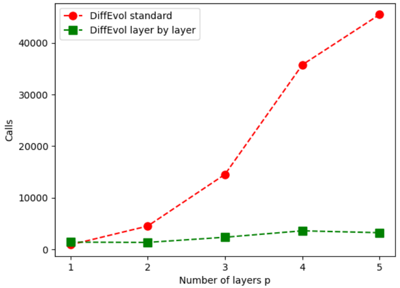


Figure: Number of calls to the objective function in standard and layer by layer optimization

- **Standard optimization**: for p layers all $\gamma_i \beta_i$, $i = 1 \ldots p$ were optimized → at the same time ($2p$ parameters)
- **Layer by layer** approach: only 2 parameters (γ_i, β_i) are optimized at a time
- Dimensionality of the solution space is reduced → **faster convergence**

Optimizing one layer at a time

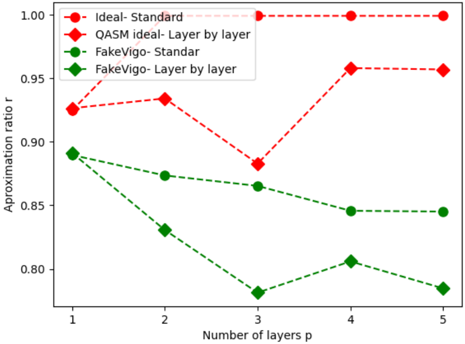


Figure: Layer by layer optmization performance

- Layer by layer does not yield the optimal result
- Number of calls to the function **drastically reduced**

Furthermore:

- **Nelder-Mead** optimizer[5]: less calls than Differential Evolution

[5] Gao, F. and Han, L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. 2012. Computational Optimization and Applications. 51:1, pp. 259-277

Running on the quantum computer

- Performance in Vigo is lower than in FakeVigo
- Optimal p is 1 as expected

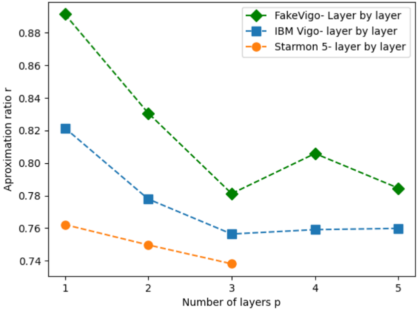


Figure: Performance on Vigo and Starmon 5

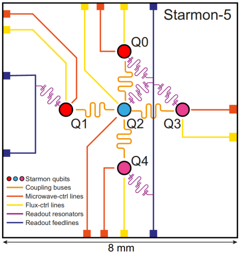


Figure: Starmon 5 chip[6]

[6] QuTech. (2018). Quantum Inspire Home. Retrieved from Quantum Inspire: https://www.quantum-inspire.com/

Outline

Conclusions and future outlook

- QAOA performance limited by the current state of the Quantum Computers
- Combinatorial optimization problems very suited for Quantum Computing
- Promising for the NISQ with reduced noise levels are more qubits

Real optimization problem example (2020)

Tail assignment problem[a]: Assigning individual aircraft to a given set of flights such that the overall cost is minimized and subject to many constraints.

- Succes probability of 96.6% for p=2 in a 2 qubit quantum computer.

[a] Improved success probability with greater circuit depth for the quantum approximate optimization algorithm. arXiv:1912.10495

Thank You!

Questions?

Code available at `https://github.com/smitchaudhary/QAOA-MaxCut`