Quantum Programming Applications: Quantum Advantage

Jens Palsberg

Jun 2, 2020

Outline

Hook: Do quantum computers already surpass classical computers for problems of practical interest? In particular, what is the status for graph problems? Google has one of the largest quantum computers, and Google people have published papers on trying to achieve a quantum advantage.

Purpose: Persuade you that quantum computers remain behind for practical problems.

Preview:
1. We will define some graph problems that are NP-complete in general.
2. We will see what Google does to implement QAOA for such problems.
3. We will look at Google's experimental results.

Transition to Body: Let us begin with a look at some problems of practical interest.

Main Point 1: We will define some graph problems that are NP-complete in general.
[Let us focus on undirected graphs]
[Each edge has a weight in $\{-1, 0, -1\}$]
[We can define graph problems that focus on minimizing a cost function]

Transition to MP2: Google wants to show that quantum computing has potential.

Main Point 2: We will see what Google does to implement QAOA for such problems.
[The Sycamore quantum computer has 54 qubits and an unusual 2-qubit gate]
[Gate selection and qubit swapping use novel ideas]
[Measurement is followed by post-processing to compensate for bias]

Transition to MP3: Now let us see how well they did.

Main Point 3: We will look at Google's experimental results.
[They repeat each experiment 50,000 times]
[For hardware subgraphs, the results are independent of the number of qubits]
[Noiseless (simulation) is much better than noisy (quantum computer)]

Transition to Close: So there you have it.

Review: Google uses just about all the implementation techniques we know. So far, the results from a quantum computer are much worse than from simulation.

Strong finish: The quest for a quantum advantage on practical problems continues. Today, graph problems; tomorrow, integer factorization; next week, machine learning.

Call to action: Join the quest for finding better ways to map optimization problems to quantum computers.

Detailed presentation

Hook: Do quantum computers already surpass classical computers for problems of practical interest? In particular, what is the status for graph problems? Google has one of the largest quantum computers, and Google people have published papers on trying to achieve a quantum advantage. One of those papers is from April 2020 and is called "*Quantum Approximate Optimization of Non-Planar Graph Problems on a Planar Superconducting Processor*", and is written by Google AI Quantum and Collaborators (many authors). The paper describes a large-scale experiment with QAOA and can be found on ArXiv, `https://arxiv.org/abs/2004.04197`. This paper uses many techniques that we have considered in this course, including a quantum algorithm, a nontrivial hybrid of classical and quantum, and an optimizing compiler.

Purpose: Persuade you that quantum computers remain behind for practical problems.

Preview:

1. We will define some graph problems that are NP-complete in general.
2. We will see what Google does to implement QAOA for such problems.
3. We will look at Google's experimental results.

Transition to Body: Let us begin with a look at some problems of practical interest.

Main Point 1: We will define some graph problems that are NP-complete in general.

[Let us focus on undirected graphs]
The paper is all about undirected graphs and the problems that we can consider for such graphs.

[Each edge has a weight in $\{-1, 0, -1\}$]
Many problems can be modeled once we assign each edge a weight in $\{-1, 0, -1\}$.

[We can define graph problems that focus on minimizing a cost function]
For a graph, we can define a cost function that works with the nodes, the edges, and the edge weights. Given such a cost function, we can define a problem that is about minimizing or maximizing the cost function. The paper considers two specific problems, namely the Sherrington-Kirkpatrick model, which uses a fully connected graph, and 3-regular MaxCut, which uses a graph that is 3-regular.

In the Sherrington-Kirkpatrick model, each edge weight is 1 or -1. We will skip the details of the cost function.

In a 3-regular graph, every node has 3 neighbors. The MaxCut problem is to find a cut whose size is at least the size of any other cut. A cut devides the nodes into two subsets and then we consider the edges that cross from one subset to the other. In MaxCut, each edge weight is 1, so the goal is to maximize the number of edges that cross from one subset of nodes to the other. MaxCut for 3-regular graphs is NP-complete.

The paper describes how to run QAOA to solve the two kinds of graph problems. QAOA is a variational algorithm that produces approximate solutions to NP-complete problems.

Transition to MP2: Google wants to show that quantum computing has potential.

Main Point 2: We will see what Google does to implement QAOA for such problems.

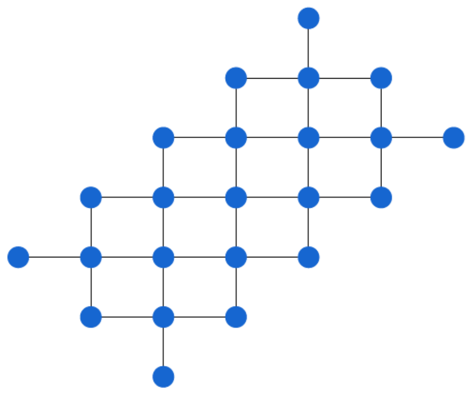
[The Sycamore quantum computer has 54 qubits and an unusual 2-qubit gate]
The Sycamore quantum computer has 54 qubits. The gate set includes an unusual 2-qubit gate called SYC.

$$
\begin{aligned}
R_{xy}(\varphi,\theta) &= \begin{pmatrix} \cos(\theta/2) & -ie^{-i\varphi}\sin(\theta/2) \\ -ie^{i\varphi}\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \\
R_z(\theta) &= e^{-i\theta Z/2} \quad = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \\
\texttt{SYC} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 0 & e^{-i\pi/6} \end{pmatrix} \\
\sqrt{\texttt{iSWAP}} &= \ldots \text{not used in this paper}
\end{aligned}
$$

The Sycamore quantum computer has low error rates compared to other quantum computers with superconducting qubits. Specifically, the SYC gate has 99.4 percent fideliy, and the measurement fidelity is 95.9 percent per qubit.

The paper uses a maximum of 23 qubits. The paper says nothing about why they stop short of using all 54 qubits. We can speculate that the compiler doesn't scale to 54 qubits; perhaps the explanation is something else.

Here is the layout of those 23 qubits.



Notice that the layout is a planar graph, that is, no edges cross each other.

Now we can consider two kinds of graphs: (1) subgraphs of the hardware graph and (2) other graphs. We should expect better performance of QAOA for subgraphs of the hardware graph; other graphs are a challenge.

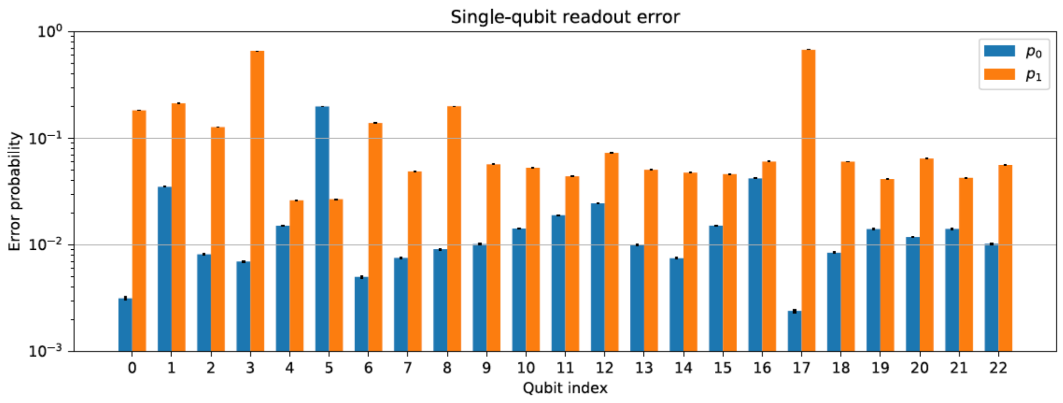
[Gate selection and qubit swapping use novel ideas]
Recall that the backend of a quantum compiler needs to do gate selection, qubit allocation, and qubit swapping. Additionally, the compiler needs to optimize for decoherence, crosstalk, and overall reliability.

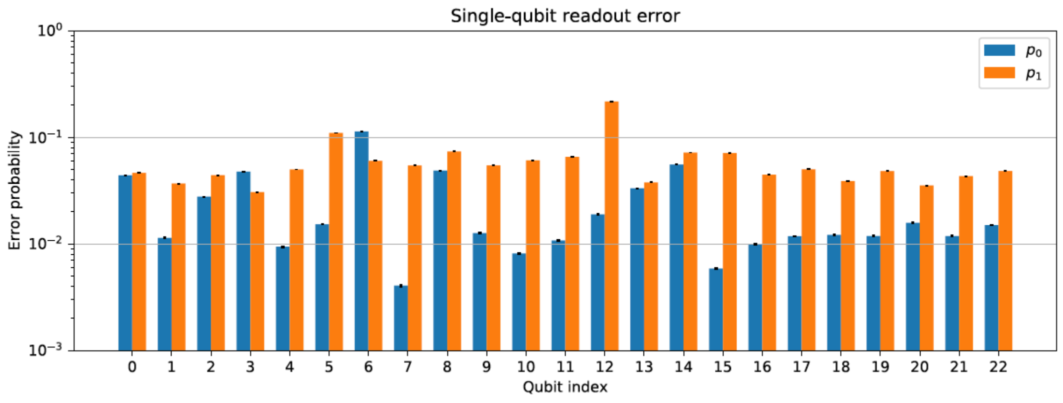
The paper says that each two-qubit gate at the source level can be mapped to two layers of SYC gates plus some single-qubit gates. In total, each source-level gate is implemented by 8 layers of SYC gates. The paper also says that for Sherrington-Kirkpatrick, each source-level gate is implemented by $3n$ layers of `SYC` gates. We will have to ask the authors how to reconcile those two counts of the `SYC` gates.

The paper uses the qubit swapping algorithm called t|ket⟩, from the paper by Cowtan et al. (2019). The Tan-Cong paper includes measurements of the t|ket⟩ algorithm. For MaxCut, the number of 2-qubit gates is quadratic in the number of 2-qubit gates for MaxCut, while the circuit depth is the same. A swap gate requires three applications of `SYC`.

The compiler does a lot of optimization. This includes optimization of adjacent 1-qubit gates. The paper observes that all R_z gates can be efficiently commuted through `SYC` and R_{xy} to the end of the circuit and discarded. This produces a circuit with alternating R_{xy} and `SYC` gates.

[Measurement is followed by post-processing to compensate for bias]
Measurement has a fairly low fidelity of 95.9 percent per qubit. Additionally, the measurement has a bias which we can model by talking about p_0 as the error that the measurement got changed from 0 to 1, and p_1 as the error that the measurement got changed from 1 to 0. The paper uses post-processing to compensate for bias in the measurements. This post-processing appears to be computationally intensive and is done as part of the classical-quantum hybrid approach.





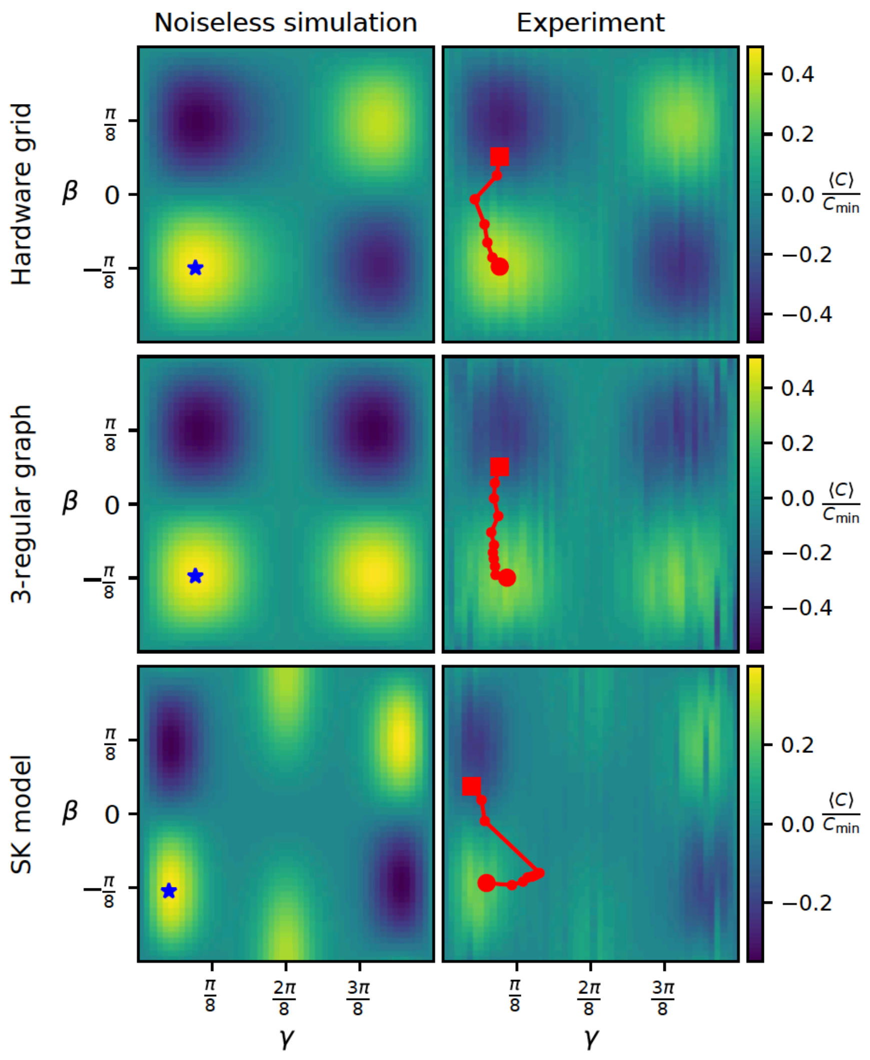
Above, the first diagram shows p_0, p_1 before post-processing, while the second diagram shows p_0, p_1 after post-processing. We see how especially p_1 is much more even across the qubits.

Transition to MP3: Now let us see how well they did.

Main Point 3: We will look at Google's experimental results.

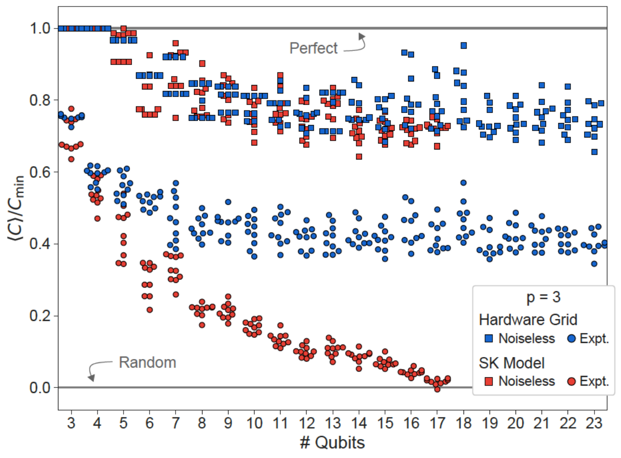
[They repeat each experiment 50,000 times]
Initially, they run QAOA with $p = 1$; later they consider $p > 1$. The QAOA classical optimizers uses Model Gradient Descent.

They estimate each expectation value by repeating the experiment 50,000 times. The real-time speed of the computation is that they measure 5,000 bitstrings per second.



Above we see a good correspondence between the experimental and the theoretical landscapes for problems of large size and complexity. Each iteration consists of 6 energy evaluations to estimate the gradient, and each energy evaluation used 25,000 circuit repetitions.

[For hardware subgraphs, the results are independent of the number of qubits]



The diagram above shows QAOA performance as a function of the number of qubits. We see that noiseless is much better than noisy, both for hardware subgraphs and for fully connected graphs. Notice that for hardware subgraphs, the results are independent of the number of qubits. The paper says that theory predicted this result: errors stay local. Notice also that for fully connected graphs, the results get worse with the number of qubits. Reason: errors propagate quickly.

[Noiseless (simulation) is much better than noisy (quantum computer)]

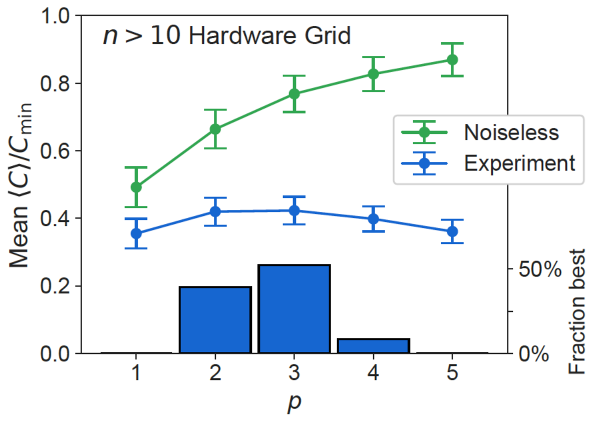


Figure 5 shows that increasing p (in QAOA) is good for noiseless but fairly neutral for noisy.

Transition to Close: So there you have it.

Review: Google uses just about all the implementation techniques we know. So far, the results from a quantum computer are much worse than from simulation.

Strong finish: The quest for a quantum advantage on practical problems continues. Today, graph problems; tomorrow, integer factorization; next week, machine learning.

Call to action: Join the quest for finding better ways to map optimization problems to quantum computers.