Optimizing quantum circuits with classical thinking

Craig Gidney
Google Quantum AI

QPL/MFPS 2018
D. Subsampling the Coefficient Oracle

In this section we introduce a technique for initializing a state with $L$ unique coefficients (provided by a classical database) with a number of $T$ gates scaling as $4L + \mathcal{O}(\log(1/\epsilon))$ where $\epsilon$ is the largest absolute error that one can
Key ideas we'll cover

1. Cost of error corrected quantum computation

2. Preparing phase-insensitive superpositions == random sampling

3. Fast proportionate sampling

4. Putting it all together for savings!
Part 1
The cost of error corrected quantum computation
"Real world" parameters: $d_{\text{code}} \approx 20$, $t_{\text{cycle}} \approx 1\text{us}$

Area: 2401 qubits

Time: 1.25$d$ cycles

2.5$d$ qubits

2.5$d$ qubits

25us
Basic error-corrected operations

- Initialization: cheap
- Measurement: cheap
- NOT gate: free
- Sqrt(NOT) gate: cheap
- Controlled-NOT: cheap
Not so cheap: \( \text{Sqrt(Sqrt(NOT))} \)

T state factory:

Footprint
\( \approx 150K \) physical qubits

Time \( \approx 150\) us

Noisy T state injections
Quantum AND gate: expensive!

\[ 0.6 \text{ms} \]

OR, NAND, NOR, etc are similarly expensive.
Wildly differing costs

Classical perspective on gate costs

Quantum perspective on gate costs

FullAdder isn't even a whole instruction.

FullAdder takes a half millisecond.
Another cost: reading data under superposition

- RAM takes $O(N)$ space to store.

- N AND gates is expensive, but N logical qubits are even more expensive.

- Instead of storing data in qubits, hardcode it into a circuit ("QROM").

- QROM circuit needs AND gates.
Reading data under superposition: QROM circuit

Encode data into presence/absence of CNOT targets.

Iterate over possible index values.
Reading data under superposition: Expensive!

Video games render frames faster than we hope to do QROM reads.

QROM query over $N$ values: $N-1$ AND gates.

Note: uncomputing AND is \(~\text{cheap}\).
Part 2
Preparing quantum states
The Preparation Problem

Given precomputed coefficients for a superposition, prepare such a superposition

\[ [a_0, a_1, a_2, \ldots, a_{N-1}] \]

\[ a_0 |0\rangle + a_1 |1\rangle + a_2 |2\rangle + \ldots + a_{N-1} |N - 1\rangle \]
Previous Approach

Set ON-vs-OFF proportion of a qubit just right with a precise rotation.

Conditioned on first qubit, set another qubit's ON-vs-OFF proportion just right.

Etc.
Cost of Previous Approach

Uses N-1 precise rotations.

Cost of precise rotation ≈ 12 AND gates. (≈50 T gates)

Roughly 3/4 of a second at N=100.
Key insight: sometimes junk is okay

You were asked to prepare a superposition:

$$\sum_k a_k |k\rangle$$

But if its usage is insensitive to phase error, you can prepare this instead:

$$\sum_k a_k |k\rangle |\text{temp}_k\rangle$$

i.e. just get the probabilities right:

$$\forall k, \langle k | \psi | k \rangle = |a_k|^2$$
Key insight: sometimes junk is okay

Context: prepared superposition is only used as a control
Key insight: sometimes junk is okay

to the system qubits
Example: Preparing

Step 1: What's the probability distribution?

\[ P(k) \propto |a_k|^2 \propto \left| \frac{1}{\sqrt{k}} \right|^2 = \frac{1}{\sqrt{k}} \]

Step 2: Create a classical sampling method.

\[ u = \text{uniform}_\text{random}() \]
\[ \text{return } u^{**2} \]

Step 3: Quantum-ify.

uniform sample \quad \text{uniform superposition}
Part 3
Sampling hard-coded probability distributions
Fitness proportionate selection

Common step in genetic algorithms

Given: a list of items with fitnesses

Goal: sample items with twice as much fitness twice as often
Common Fitness-Proportionate Selection Methods

https://jbn.github.io/fast_proportional_selection/

<table>
<thead>
<tr>
<th>Method</th>
<th>Classical Sampling Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Walk</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Bisecting Search</td>
<td>$O(lg N)$</td>
</tr>
<tr>
<td>Stochastic Acceptance</td>
<td>$O(p_{max} N)$</td>
</tr>
</tbody>
</table>

![Image showing fitness values]

$f_a = 7$, $f_b = 4$, $f_c = 3$, $f_d = 1$, $f_e = 5$
# Common Fitness-Proportionate Selection Methods

https://jbn.github.io/fast_proportional_selection/

<table>
<thead>
<tr>
<th>Method</th>
<th>Classical Sampling Cost</th>
<th>Quantum Preparation Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Walk</td>
<td>$O(N)$</td>
<td>$O(N \lg(1/\varepsilon))$</td>
</tr>
<tr>
<td>Bisecting Search</td>
<td>$O(\lg N)$</td>
<td>$O(N \lg(1/\varepsilon))$</td>
</tr>
<tr>
<td>Stochastic Acceptance</td>
<td>$O(p_{\text{max}}N)$</td>
<td>Not Reversible</td>
</tr>
</tbody>
</table>

Search trees don't help quantum cost. Under superposition, you must do the operations for **every** path.
## Common Fitness-Proportionate Selection Methods

[https://jbn.github.io/fast_proportional_selection/](https://jbn.github.io/fast_proportional_selection/)

<table>
<thead>
<tr>
<th>Method</th>
<th>Classical Sampling Cost</th>
<th>Quantum Preparation Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Walk</td>
<td>$O(N)$</td>
<td>$O(N \lg(1/\varepsilon))$</td>
</tr>
<tr>
<td>Bisecting Search</td>
<td>$O(\lg N)$</td>
<td>$O(N \lg(1/\varepsilon))$</td>
</tr>
<tr>
<td>Stochastic Acceptance</td>
<td>$O(p_{\text{max}} N)$</td>
<td>Not Reversible</td>
</tr>
<tr>
<td>Alias Sampling*</td>
<td>$O(1)$</td>
<td>$O(N + \lg(1/\varepsilon))$</td>
</tr>
</tbody>
</table>

*Walker 1974: "New fast method for generating discrete random numbers with arbitrary frequency distributions"
Alias sampling: repacking histograms

Pick initial item uniformly at random, then probabilistically switch to an alternate item.
How to repack a histogram

Average
How to repack a histogram
How to repack a histogram

The diagram illustrates how to repack a histogram. It shows an initial distribution (left) that is too large and needs to be repacked (middle) to achieve the desired distribution (right). The process involves redistributing the bars to match the target distribution, ensuring that the total number of observations remains unchanged.
How to repack a histogram

top up by transferring
How to repack a histogram
How to repack a histogram

It's okay to undershoot the average when donating.
How to repack a histogram
How to repack a histogram
How to repack a histogram
Repacking costs

Linear time using Vose's algorithm

Doesn't affect runtime of quantum algorithm (classically precomputed)

All approximations happen here. Sampling adds zero additional error!
Part 4
Putting it all together
Using alias sampling to prepare a superposition

**Classical Sampling**

```python
def alias_sample(alternates, keep_weights, precision):
    # Pick an item uniformly at random.
    n = len(alternates)
    k = randint(n)
    # Look up alternate item and keep chance.
    alt = alternates[k]
    keep = keep_weights[k]
    # Potentially switch to alternate item.
    threshold = randint(2**precision)
    kept = threshold < keep
    return k if kept else alt
```

**Quantum Preparation**
Cost of alias preparation

Preparing a uniform superposition costs $O(\lg N + \lg 1/\varepsilon)$

QROM lookup uses **N-1 AND gates** (dominant cost)

Compare+swap costs $O(\lg N + \lg 1/\varepsilon)$

Runs at ≈20Hz given N=100.

(an order of magnitude faster)
Part 5
Wrap-up
What we covered: section 3-D of arXiv:1805.03662

Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity

Ryan Babbush,¹, * Craig Gidney,² Dominic W. Berry,³ Nathan Wiebe,⁴ Jarrod McClean,¹ Alexandru Paler,⁵ Austin Fowler,² and Hartmut Neven¹

[...]

D. Subsampling the Coefficient Oracle

In this section we introduce a technique for initializing a state with $L$ unique coefficients (provided by a classical database) with a number of T gates scaling as $4L + \mathcal{O}(\log(1/\epsilon))$ where $\epsilon$ is the largest absolute error that one can

[...]
Preparation is a small part of a larger algorithm

This talk

Quantum phase estimation

This talk

\[ N^{\lg N} \]
Estimated costs of the overall algorithm

<table>
<thead>
<tr>
<th>problem</th>
<th>physical qubits</th>
<th>execution time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p = 10^{-3}$</td>
<td>$p = 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>$p = 10^{-3}$</td>
<td>$p = 10^{-4}$</td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hubbard model</td>
<td>72</td>
<td>$1.7 \times 10^6$</td>
</tr>
<tr>
<td>Hubbard model</td>
<td>128</td>
<td>$2.4 \times 10^6$</td>
</tr>
<tr>
<td>Hubbard model</td>
<td>200</td>
<td>$3.8 \times 10^6$</td>
</tr>
<tr>
<td>Hubbard model</td>
<td>800</td>
<td>$1.5 \times 10^7$</td>
</tr>
<tr>
<td>Electronic structure</td>
<td>54</td>
<td>$1.7 \times 10^6$</td>
</tr>
<tr>
<td>Electronic structure</td>
<td>128</td>
<td>$2.9 \times 10^6$</td>
</tr>
<tr>
<td>Electronic structure</td>
<td>250</td>
<td>$5.1 \times 10^6$</td>
</tr>
<tr>
<td>Electronic structure</td>
<td>1024</td>
<td>$2.3 \times 10^7$</td>
</tr>
</tbody>
</table>

Contrast with previous work*, which had:
- Execution times in months
- Using hundreds of millions of physical qubits
- Assuming 10 \text{nanosecond} T gates instead of 150us T gates

*Reiher et al: "Elucidating reaction mechanisms on quantum computers"
Key Takeaways

- Quantum algorithms start with a constant factor penalty of a billion (if not more).

- When a quantum subroutine is phase-insensitive, try porting classical methods.

- Random sampling methods seem to port particularly well.

- Alias sampling dominates bisecting search sampling yet is less well known.
Thanks for listening!