The Bitter Truth About Quantum Algorithms in the NISQ Era

(TU Wien, November 24th, 2021)

Prof. Dr. Dr. h.c. Frank Leymann
Kurt Gödel Visiting Professor TU Wien
Institut für Architektur von Anwendungssystemen (IAAS)
Universität Stuttgart

Leymann, Frank; Barzen, Johanna: The bitter truth about gate-based quantum algorithms in the NISQ era.
In: Quantum Science and Technology, 2020
Technological Problems

Decoherence: Qbits are not stable
⇒ State of a qbit decays over time (often, rather quick!)
⇒ Implementation of qbits disturb each other
⇒ Increasing number of qbits is quite difficult

Gate Infidelity: Each operation is (a bit) imprecise
⇒ Error of an algorithm increases with number of operations
⇒ Only algorithms with "few" operations can be executed precisely

Readout Error: Measurement of a qbit is imprecise
⇒ Results are distorted

Qbit Connectivity: Not all qbits are physically connected
⇒ 2-qbit operations cannot be applied to arbitrary pairs of qbits
   → Reminder: 2-qbit operations are mandatory in a set of universal operations
⇒ Additional SWAP operations must be performed
⇒ Number of operations of proper algorithms further limited
Decoherence
For $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ there is a $\theta \in [0, \pi]$ and a $\rho \in [0, 2\pi]$, such that

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\rho} \sin \frac{\theta}{2} |1\rangle$$

$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\rho} \sin \frac{\theta}{2} |1\rangle \Leftrightarrow (\theta, \rho)$
Decoherence

\[ T_1 \ (\text{relaxation time}) \ - \text{collapse:} \]
Transition into an orthogonal state

\[ T_2 \ (\text{dephasing time}) \ - \text{small disturbance:} \]
Random change of phase
Non-Applicability of Classical Error Correction

Redundant codes (copies of qbits) cannot be created: **No-Cloning**!

A qbit will not change in a discrete manner (0 to 1, 1 to 0), but the amplitudes of superposition can be changed arbitrarily: **Continuous Errors**!

Reading means measurement, but this destroys the state, i.e. recovery of the original state is impossible: **Destructive Reads**!
Physical/Logical Qbits

Encoding 1 qbit by 9 qbits allows to detect and correct any (bit single) error!

\[
|0\rangle \mapsto \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}
\]

\[
|1\rangle \mapsto \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}
\]

...and other encodings are possible. But:

Multiple noisy "physical" qbits needed to realize 1 stable "logical" qbit!
Error Correction of Qbits

|ψ⟩  "Noise"
|0⟩
|0⟩  Syndrome Recognition
|0⟩  Recovery
|0⟩  Error Correction (EC)

Encoding
Gate Fidelity
1-Qbit Operators: Decomposition

A set $\mathcal{U}$ of 1-qbit operators is called *universal* $\iff$
Each 1-qbit operator is a finite combination of operators from $\mathcal{U}$

Let $U$ be a 1-qbit operator. Then:

$$\exists \alpha, \beta, \gamma, \delta \in \mathbb{R} : U = e^{i\alpha} R_z (\beta) R_y (\gamma) R_z (\delta)$$
Gates are Inherent Imprecise

\[ R_x(\theta) \] is rotation by angle \( \theta \) around x-axis

Exact rotation around an angle is in general impossible

\[ \Rightarrow \text{Rotation is inherent imprecise} \]

\[ \Rightarrow \text{Each qbit operation } U = e^{i\alpha R_z(\beta)} R_y(\gamma) R_z(\delta) \text{ has a small error} \]
Gate Errors

Applying the algorithm $U_T \circ \cdots \circ U_1$ to $\varphi_0$ results in $\varphi_T$:

$$|\varphi_T\rangle = U_T \circ \cdots \circ U_1 |\varphi_0\rangle$$

Each operation $U_i$ is a bit imprecise, produces a small deviation from the exact result, i.e. instead of $U_i$ an operation $\tilde{U}_i$ is performed:

Thus, instead of $|\varphi_1\rangle = U_1 |\varphi_0\rangle$ the result $\tilde{U}_1 |\varphi_0\rangle = |\varphi_1\rangle + |E_1\rangle$ is produced (Gate Error or [lack of] Gate Fidelity)

I.e. the final computed result of the algorithm is:

$$|\tilde{\varphi}_T\rangle = |\varphi_T\rangle + |E_T\rangle + \tilde{U}_T |E_{T-1}\rangle + \tilde{U}_T \tilde{U}_{T-1} |E_{T-2}\rangle + \cdots + \tilde{U}_T \tilde{U}_{T-1} \cdots \tilde{U}_2 |E_1\rangle$$
Error Propagation

\begin{equation}
\left\| \left| \tilde{\phi}_T \right\rangle - \left| \phi_T \right\rangle \right\| \leq \left\| \left| E_T \right\rangle \right\| + \left\| \left| E_{T-1} \right\rangle \right\| + \left\| \left| E_{T-2} \right\rangle \right\| + \ldots + \left\| \left| E_1 \right\rangle \right\|
\end{equation}

Let \( \varepsilon \) be the maximum error of all gates:

\[ \forall 1 \leq t \leq T: \left\| (\tilde{U}_t - U_t) \right\| \leq \varepsilon \]

\[ \Rightarrow \left\| \left| \tilde{\phi}_T \right\rangle - \left| \phi_T \right\rangle \right\| \leq T \varepsilon \]

The accumulated error grows linear with the length of the computation.
Threshold Theorem

For any required precision of a computation C of a set of ideal gates, there is an implementation C’ based on fault tolerant gates that computes the results of C within the required precision...

...if the fault tolerant gates fail less than a threshold η-times

Today(*) (2019), η≈10^{-2}

⇒ Fault tolerance scales - in principle!

Noisy Intermediate Scale Quantum computing: NISQ

Fault-Tolerance: Principle

1 Qbit is encoded by k error-correcting Qbits

Block, physical Qbits

Universal gate G is substituted by a *coded gate* G’
(coded gate G’ ist quantum subroutine implementing the functionality of G)

After executing a coded gate, error correction on affected blocks are run
Implication of Noise

N noisy "physical" qbits are needed to realize 1 stable "logical" qbit!

$\Rightarrow$ More qbits needed than estimated by theoretical algorithms

Single universal but noisy gate is realized by quantum subroutine!

Error correction on noisy qbit is run periodically!

$\Rightarrow$ More operations needed than estimated by theoretical algorithms
Metrics of an Algorithm
Depth and Width of an Algorithm

The depth of a quantum circuit is the number of layers of 1- or 2-qbit gates that operate in parallel on disjoint qbits.

The width of a quantum circuit is the number of manipulated qbits.
Examples

Depth(G) = 2
Width(G) = 7

Depth(G') = 3
Width(G') = 3

© Frank Leymann
Noisy Algorithms

Rough estimation of the "size" of a quantum algorithm that can be performed without errors:

\[ wd \ll \frac{1}{\varepsilon} \]

w: width
d: depth
\( \varepsilon \): error rate
Consequences

\[ wd \ll \frac{1}{\varepsilon} \]

*Deep quantum algorithms* ⇒ few qbits
⇒ efficient classical simulation possible

*Shallow quantum algorithms* ⇒ many qbits
⇒ potential for *quantum advantage*
Transpilation
(a.k.a. Cross-Compilation)
Transpilation: Mapping to Hardware Gates

Original circuit

Transpiled circuit

Original circuit

Transpiled circuit

OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
creg c[1];
x q[0];
h q[0];
cx q[0],q[1];
measure q[0] -> c[0];
Transpilation: Increasing Depth

Original circuit

Transpiled circuit

```openqasm
OPENQASM 2.0;
include "qelib1.inc";
qreg q[5];
creg c[3];
u2(0, 3.141592653589793) q[0];
u2(0, 3.141592653589793) q[1];
cx q[0], q[1];
cx q[1], q[0];
cx q[0], q[1];
u2(0, 3.141592653589793) q[2];
cx q[1], q[2];
cx q[0], q[1];
cx q[1], q[0];
cx q[0], q[1];
cx q[1], q[2];
```
Transpilation: Decreasing Depth

Original circuit

Transpiled circuit
Transpilation

Original circuit

Transpiled circuit

...on simulator

...on QPU
Circuit Rewrite: Implications

The depth of a circuit can often be reduced by "shifting gates to the left as far as possible", i.e. without sacrificing the data flow. This is mainly hardware independent.

Hardware dependent rewrite is required, e.g. to map the gates of a hardware-independent circuit to the gates supported by the concrete hardware. This typically increases the depth of an algorithm (but may decrease it). ⇒ Inspection of transpiled circuit needed to assess executability.
Input Preparation
Reminder: Quantum Algorithm

- State Preparation
- Unitary Transformation
- Measurement
- Postprocessing

Preprocessing
Quantum Algorithm: Paper Version

Quantum algorithm for linear systems of equations

Arun T. Harrow,1, Avinatan Hassidim,1,2 and Seth Lloyd3

1Research Laboratory for Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
2Department of Mathematics, University of Toronto, Toronto, ON M5S 3G3, Canada
3Department of Physics and Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, CA 91125, USA

Quantum computers are devices that perform computations in ways that classical computers cannot. For certain problems, quantum computers can solve them efficiently. However, in some cases, simulations on classical computers can be much faster than the calculations based on quantum computers. In this paper, we present a novel algorithm for solving systems of linear equations, which we call the quantum algorithm for linear systems of equations (QALSE). The QALSE algorithm is designed to be implemented on a quantum computer and can solve linear systems of equations exponentially faster than any known classical algorithm.

1. Introduction

Quantum computers are devices that perform computations in ways that classical computers cannot. For certain problems, quantum computers can solve them efficiently. However, in some cases, simulations on classical computers can be much faster than the calculations based on quantum computers. In this paper, we present a novel algorithm for solving systems of linear equations, which we call the quantum algorithm for linear systems of equations (QALSE). The QALSE algorithm is designed to be implemented on a quantum computer and can solve linear systems of equations exponentially faster than any known classical algorithm.

Quantum Algorithm

Next, we apply the conditional Hadamard-evolution \( \sum_{i=0}^{2^n-1} |i\rangle \langle i| e^{-iH\pi/2^n} \) on \( |\psi\rangle = |\psi(0)\rangle \), where \( \psi(0) = O|\psi(0)\rangle \).

To factorizing the first register, given the state:

\[
\sum_{i=0}^{2^n-1} m_i |i\rangle |\beta_i\rangle
\]

where \( |\beta\rangle \) is the Fourier basis states, and \( |\beta_i\rangle \) is large if and only if \( i = \beta \). Thus, \( |\beta\rangle = M^{-1} |\beta_0\rangle \), so we can replaced \(|\beta\rangle \) state to obtain:

\[
\sum_{i=0}^{2^n-1} m_i |i\rangle |\beta_i\rangle
\]

Adding an oracle trick and rotating conditioned on \( |\beta_i\rangle \) yields

\[
\sum_{i=0}^{2^n-1} m_i |i\rangle |\beta_i\rangle n\left( a - \frac{C}{2^n}\right)\mbox{.}
\]

where \( C = O(1)\). We now make the phase estimation to uncompute the \( |\beta_i\rangle \). If the phase estimation were perfect, we would have \( n = 1 \). If \( n = \beta \), we otherwise. Assuming this new, we obtain

\[
\sum_{i=0}^{2^n-1} m_i |i\rangle \left( \sqrt{1 - \frac{C}{2^n}} + \frac{C}{2^n} \right)\mbox{.}
\]

To finish the inversion we measure the last qubit. Conditioned on seeing 1, we have the state

\[
\sum_{i=0}^{2^n-1} \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} C_{i,j} |j\rangle|\beta_i\rangle
\]

which corresponds to \( |\alpha\rangle = \sum_{i=0}^{2^n-1} \beta_i |i\rangle \) up to normalization. This can determine the normalization factor from the probability of obtaining 1. Finally, we make a measurement \( M \) whose expectation value \( \langle \beta | M | \beta \rangle \) corresponds to the feature of \( M \) that we wish to evaluate.

© Frank Leymann
Data for the Algorithm

We also need an efficient procedure to prepare $|b\rangle$.

Diagram:

- Data as Quantum State
- Unitary Transformation
- Data Points (Vectors)
- Set of Vectors
- Matrices
- Categorical Data
- QUBOs

Assumption
Data as Quantum State

\[ (x_1, \ldots, x_N) \mapsto \left( \begin{array}{c} \cos x_1 \\ \sin x_1 \end{array} \right) \otimes \cdots \otimes \left( \begin{array}{c} \cos x_N \\ \sin x_N \end{array} \right) \]

© Frank Leymann
State Preparation

Various possibilities (each with pros and cons), e.g.:

- Basis Encoding
- Amplitude Encoding
- Tensor product encoding
- Schmidt encoding

Corresponds to two categories

- Digital encoding
  - …for performing arithmetics
- Analog encoding
  - …for processing in high-dimensional feature spaces
Basis Encoding

Let \( x \in \mathbb{N} \)

Then, \( x \) will be binary encoded, i.e. \((x_1, \ldots, x_n) \in \{0,1\}^n \) with \( \sum x_k 2^k = x \)

\( x \mapsto |x_1, \ldots, x_n\rangle \) is called *Basis Encoding* of \( x \in \mathbb{N} \)

Base encoding is a representative of *digital encodings*
Basis Encoding: Circuit

Resources for encoding n bits:
- n qbits
- n gates
- depth 1
- 0 ancillae

Obviously, this circuit can be generated in a preprocessing step:

\[ X^{b_1} \otimes \cdots \otimes X^{b_n} |0\cdots0\]
Basis Encoding: Real Numbers

A number $x \in \mathbb{R}$ is approximated in binary representation to $k$ decimal places:

$$x \approx \sum_{i=0}^{n} b_i 2^i + \sum_{i=1}^{k} b_{-i} \cdot \frac{1}{2^i}$$

E.g. let $x=1.7$ and $k=4$, i.e. $x = 1 \cdot 2^0 + \sum_{i=1}^{4} b_{-i} \cdot \frac{1}{2^i}$

...next, compute decimal places:

<table>
<thead>
<tr>
<th>Calculation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.7 \cdot 2$</td>
<td>$1.4$</td>
</tr>
<tr>
<td>$0.4 \cdot 2$</td>
<td>$0.8$</td>
</tr>
<tr>
<td>$0.8 \cdot 2$</td>
<td>$1.6$</td>
</tr>
<tr>
<td>$0.6 \cdot 2$</td>
<td>$1.2$</td>
</tr>
<tr>
<td>$0.2 \cdot 2$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>$0.4 \cdot 2$</td>
<td>$...$</td>
</tr>
</tbody>
</table>

$0.7_{10} = 10110_2$  
...i.e. $1.7$ approximated to 4 decimal places: $11011$
Input Preparation of Real Numbers: Base Encoding

State Preparation

|0⟩ --- $X^{b_n}$ --- |b_n⟩

⋯

|0⟩ --- $X^{b_{-k}}$ --- |b_{-k}⟩

Preprocessing

Compute binary representation of x

$$x \approx \sum_{i=0}^{n} b_i 2^i + \sum_{i=1}^{k} b_{-i} \cdot \frac{1}{2^i}$$

Generate corresponding circuit

$$X^{b_n} \otimes \cdots \otimes X^{b_{-k}} |0\cdots0⟩$$

(Note: signs w.l.o.g. not considered)
Basis Encoding of Real Vectors

The sign of a number is represented by a leading 1 ("−") or 0 ("+")

I.e. (4 decimal places): \(-0.7_{10} = 1\ 1011_2\ +0.1_{10} = 0\ 1001_2\ +0.2_{10} = 0\ 0011_2\)

Thus,

\[
x = \begin{pmatrix} -0.7 \\ 0.1 \\ 0.2 \end{pmatrix} \mapsto \begin{pmatrix} 11011 \\ 01011 \\ 00011 \end{pmatrix} \mapsto \left| 11011\ 01001\ 00011 \right\rangle = \left| x \right\rangle
\]

(It’s obvious how to generalize the preparation method for real numbers in base encoding to real vectors in base encoding)
Basis Encoding
of Data Sets

Let $D = \{x_1, \ldots, x_m\}$ be a data set to be processed by a quantum algorithm.

Representation of D as a quantum state:

$$|D\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |x_i\rangle$$

Example: $x_1 = |101\rangle$ and $x_2 = |011\rangle$, then

$$|D\rangle = \frac{1}{\sqrt{2}} \left( |101\rangle + |011\rangle \right)$$

...as a amplitude vector:

$$|D\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

I.e. state vectors of binary data sets are typically sparse vectors.
Amplitude Encoding

Let $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$ be a unit-length vector, $N = 2^n$ ($\Rightarrow |x_i| \leq 1 \ \forall i$)

$x \mapsto \Sigma x_i|i\rangle$ is called amplitude encoding of $x \in \mathbb{R}^N$

The amplitude encoding is an analog encoding

For $x \in \mathbb{R}^N$, $N \neq 2^n$, use a proper embedding (called padding):

$x \mapsto (x,0) \in \mathbb{R}^N \times \mathbb{R}^M$, $(N+M) = 2^n$, for the smallest possible $n$
Amplitude Encoding of Non-Unit-Length Vectors

For \( x \in \mathbb{R}^N \setminus \{0\} \) the encoding is

\[
x \mapsto \sum \frac{x_i}{\| x \|} \left| i \rightangle
\]

**Note:** a matrix \( A \in \mathbb{R}^{n \times m} \) can be represented as vector in \( \mathbb{R}^{nm} \)

\[
\left| A \rightangle = \sum \frac{a_{ij}}{\| A \|} \left| i \rightangle \left| j \rightangle
\]

The \( \| \cdot \| \) may be computed classically as preprocessing step
Normalization and "Neighborhood"

Normalizing the members set $D \subseteq \mathbb{R}^N$ changes the relation between the members

...which must be considered in certain algorithms (e.g. clustering)
(Tensor) Product Encoding

Let \( x = (x_1, \ldots, x_N) \in \mathbb{R}^N \) be a unit-length vector (\( \Rightarrow |x_i| \leq 1 \ \forall i \))

Each \( x_i \) is represented by a separate qbit:

\[
    x_i \mapsto \cos x_i \cdot |0\rangle + \sin x_i \cdot |1\rangle
\]

Then, \( x \mapsto \left( \begin{array}{c} \cos x_1 \\ \sin x_1 \end{array} \right) \otimes \cdots \otimes \left( \begin{array}{c} \cos x_N \\ \sin x_N \end{array} \right) \)

is called \( \text{(tensor) product encoding} \) of \( x \) (a.k.a. \text{angle encoding})

Product encoding is a representative of an \text{analog encoding}
Circuit for Product Encoding

\[ R_y(2x) = \begin{pmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{pmatrix} \Rightarrow R_y(2x) |0\rangle = \cos x \cdot |0\rangle + \sin x \cdot |1\rangle \]

Thus:

\[ \left( \bigotimes_{i=1}^{n} R_y(2x_i) \right) |0\cdots0\rangle = \left( \begin{pmatrix} \cos x_1 \\ \sin x_1 \end{pmatrix} \right) \otimes \cdots \otimes \left( \begin{pmatrix} \cos x_n \\ \sin x_n \end{pmatrix} \right) \]
Input Preparation of Real Vectors: Product Encoding

Preprocessing

State Preparation

Generate the circuit

$$\left( \bigotimes_{i=1}^{n} R_y(2x_i) \right) |0\cdots0\rangle$$
Let $x \in V \otimes W$. There exist ONB $\{u_j\} \subseteq V$ and $\{v_j\} \subseteq W$ such that:

$$x = \sum_{i=1}^{K} \lambda_i \cdot u_i \otimes v_i$$

mit $\lambda_i > 0$ and $\sum \lambda_i = 1$.

$\lambda_i$ are called \textit{Schmidt Coefficients} of $v$, $K$ is called \textit{Schmidt Number} of $v$ (a.k.a.: \textit{Schmidt Rank})
Schmidt Decomposition via Singular Value Decomposition

Split the quantum register $R$ into two parts: $R = V \otimes W$

Choose ONB $\{e_i\}$ and $\{f_j\}$ for $V$ and $W$

Represent $x$ as $x = \sum_{i,j} \beta_{ij} \cdot e_i \otimes f_j$

Compute the singular value decomposition of $M = (\beta_{ij})$: $M = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} A \\ 0 \end{pmatrix} V^*$

Choose the column vectors of $U_1 \rightarrow \{u_1, \ldots, u_K\}$

Choose the column vectors of $V \rightarrow \{v_1, \ldots, v_K\}$

$A = \text{diag}(\lambda_1, \ldots, \lambda_K)$

$\Rightarrow x = \sum_{i=1}^{K} \lambda_i \cdot u_i \otimes v_i$
State Preparation Based On Schmidt Decomposition

\[ x = \sum_{i,j} \beta_{ij} \cdot e_i \otimes f_i, \text{ then SVD: } (\beta_{ij}) = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} A \\ 0 \end{pmatrix} V^* \]

\[ |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \]

\[ \{ |x\rangle \} \]

\[ U_1, V, A \text{ have to be composed of 1-qbit & 2-qbit operations:} \]

\[ \rightarrow U_1 = M_1 \otimes \ldots \otimes M_r, \quad V = M_{r+1} \otimes \ldots \otimes M_{r+s}, \quad A = M_{r+s+1} \otimes \ldots \otimes M_{r+s+t} \]

where each \( M_i \) is a 1-qbit gate or a CNOT

\[ \ldots \text{and each of the 1-qbit gates is represented as rotations:} \]

\[ M_j = e^{i\alpha_j} R_z(\beta_j) R_y(\gamma_j) R_z(\delta_j) \]
Refinement

\[ |0\rangle \rightarrow \underbrace{A} \rightarrow \underbrace{U_1} \rightarrow \underbrace{V} \rightarrow \{x\rangle \]

\[ e^{i\alpha_j I} R_z(\beta_j) R_y(\gamma_j) R_z(\delta_j) \]

\[ |0\rangle |0\rangle |0\rangle |0\rangle |0\rangle |0\rangle |0\rangle |0\rangle \]

\[ M_j \quad M_i \quad \ldots \quad M_k \quad M_s \quad M_r \quad M_a \quad M_b \quad M_c \]
Input Preparation of Real Vectors: Schmidt Decomposition

State Preparation

\[ A = \bigotimes_i M_i, \quad U_1 = \bigotimes_j M_j, \quad V = \bigotimes_k M_k \]

Preprocessing

Compute \( A, U_1, V \)

Compute \( M_l = e^{i\alpha_l} R_z(\beta_l) R_y(\gamma_l) R_z(\delta_l) \)

Generate corresponding circuits
General Proceeding: Input Preparation

Preprocessing
- Decide encoding
- Compute parameter
- Generate gates
- Generate circuit

State Preparation
- Execute circuit

Unitary Transformation

Measurement

Postprocessing
Implication of State Preparation

State preparation requires additional operations and additional qbits as well as classical preprocessing compared to the "ideal" algorithm.
Oracle Expansion
Reminder: Quantum Algorithm

- State Preparation
- Preprocessing
- Unitary Transformation
- Measurement
- Postprocessing
- Result
Algorithm of Deutsch

\[
\begin{align*}
|0\rangle & \quad \text{H} \quad \text{U}_f \quad \text{H} \quad |x\rangle \\
|1\rangle & \quad \text{H} \quad \text{U}_f
\end{align*}
\]
Sample Oracle

Let \( f : \{0,1\} \rightarrow \{0,1\} \) be the function \( 0 \mapsto 1, 1 \mapsto 0 \)

For \( U_f|x,y\rangle = |x, y\oplus f(x)\rangle \) an oracle is:

\[
\begin{array}{ccc}
0 & \rightarrow & 0 \\
0 & \rightarrow & 1 \\
0 & \rightarrow & 0 \\
1 & \rightarrow & 1 \\
1 & \rightarrow & 1
\end{array}
\]

(Note: different \( f \) require different \( U_f \))
Resulting Circuit

\[ |0\rangle \quad \text{H} \quad \text{U}_f \quad \text{H} \quad |x\rangle \]

\[ |1\rangle \quad \text{H} \]

(Oracle Expansion)

\[ |0\rangle \quad \text{H} \quad \text{X} \quad \text{X} \quad \text{H} \quad |x\rangle \]

\[ |1\rangle \quad \text{H} \quad \text{U}_f \quad |y \oplus f(x)\rangle \]
Algorithm of Shor

\[ |0\rangle \xrightarrow{H} |0\rangle \]
\[ |0\rangle \]
\[ |0\rangle \xrightarrow{H} |0\rangle \]
\[ |0\rangle \]
\[ |0\rangle \]
\[ |0\rangle \]
\[ |0\rangle \]

...computing \( f(x) = a^x \mod n \) (\( \rightarrow \) multiplication, addition,\( \ldots \))

Classical Post-Processing (continued fractions)
Addition

\[ R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \] (and \( R_0 = Z \))

Time complexity: \( O(n^2) \)

(Depth of the circuit can be significantly reduced, e.g. \( Z^C \) of \( S_2 \) can run in parallel to \( Z^C \) of \( S_1 \) etc…)

QFT
Multiplication: Sample

Computation of $b+ax$, $a$: 3-bit constant, $x$: 3 qbit, $b$: 6 qbit

Shor Circuit: Summary
Oracle expansion requires additional operations (and additional qbits) compared to the "ideal" algorithm.
Connectivity
Reminder: Quantum Algorithm

State Preparation → Unitary Transformation → Measurement

Preprocessing

Postprocessing → Result
The set of 1-qbit Operators and CNOT is universal.
Hardware Restrictions

2-qbit operator on two qbits requires connection between them

⇒ Connectivity of a quantum chip is important
Hardware: Connectivity

IBM’s 10 Quantum Device Lineup


http://docs.rigetti.com/en/1.9/_images/acorn.png
Swap Operator

SWAP : $\mathbb{H} \otimes \mathbb{H} \rightarrow \mathbb{H} \otimes \mathbb{H}$

I.e. both input qbits are exchanged

$|00\rangle \leftrightarrow |00\rangle$
$|01\rangle \leftrightarrow |10\rangle$
$|10\rangle \leftrightarrow |01\rangle$
$|11\rangle \leftrightarrow |11\rangle$

SWAP Gate
Example: Considering Topology

Logical Algorithm

\[ b_1 \quad b_2 \quad b_3 \quad b_4 \]

\[ R_x \]

Topology Graph

Initially, \( b_i \mapsto q_i \) ("qbit allocation")

Physical Algorithm
Example: Variation-Aware Qbit Movement

Typically, 2-qbit operations along different connections have different success rates.

Annotation $s_{ij}$ on the edge $\{q_i, q_j\}$ denotes the success rate of a 2-qbit operation involving qbit $q_i$ and $q_j$.

Scenario: a 2-qbit operation $\Omega$ is to be performed on $q_1$, $q_3$.

Swapping $q_3 \rightarrow q_2$, followed by $\Omega(q_1,q_2)$ has success rate $0.3 \times 0.5 = 0.15$.

Swapping $q_3 \rightarrow q_4 \rightarrow q_5 \rightarrow q_6 \rightarrow q_7 \rightarrow q_8$ followed by $\Omega(q_1,q_8)$ has success rate $0.8 \times 0.8 \times 0.9 \times 0.9 \times 0.7 \times 0.9 = 0.33$.

⇒ Using a single SWAP followed by $\Omega$ has a lower success rate than using 5 SWAPs followed by $\Omega$.

⇒ Success rate of qbit connections influences the number of SWAPs performed as well as error rates of 2-qbit operations.

Even worse, the success rate changes over time!
Example: Variation-Aware Qbit Allocation

The qbits of the quantum circuit must be assigned to physical qbits of the QPU
- This is an initial allocation that changes during the execution
- The goal is to improve reliability of the computation

Naive allocation selects any subgraph to minimize SWAPs
Considering success rate of connections determines connected subgraph with maximum weights
- In the example: Q0, Q1, Q2 \(\mapsto\) q5, q6, q7

```
qreg Q[3];
creg C[3];

x Q[0];
cx Q[0], Q[1];
cx Q[2], Q[1];
measure Q[1] \rightarrow\ C[1];
```
Implication of Connectivity

The connectivity of a QPU implies the injection of additional (SWAP) operations into the "ideal" algorithm.

The success rate of qbit connections influence the number of SWAPs as well as the error rate of 2-qbit operations.

Considering the success rate of qbit connections as well as error rate of 1-qbit operations during qbit allocation of a quantum circuit influences the reliability of its execution.
Readout Errors
Reminder: Quantum Algorithm
Readout Errors

Duration of a measurement is significantly larger than decoherence time

⇒ a qbit under measurement may relax during this time
  (e.g. flip from $|1\rangle$ to $|0\rangle$ in between)

Thus, *readout errors* correspond to
disturbed probability distributions of measured results
Let $t$ be the true distribution, $m$ be the measured distribution — $t, m \in \mathbb{N}^k$ — where $k$ is the number of values, and $t_i, m_i \in \mathbb{N}$ is the count of the $i$-th value.

$t, m$ are related by a *calibration matrix*\(^(*)\) $C$: $t = C \cdot m$ with $C_{ij} = \text{Prob}(\text{measured value} = j \mid \text{true} = i)$

**Correcting readout errors** means determining the calibration matrix $C$ *(unfolding method)*

\(^(*)\) a.k.a. *response matrix*
Constructing the Calibration Matrix

Construct and measure each element of the computational basis $|i\rangle \in \{0,1\}^n$

- I.e. use the above so-called *calibration circuits* $C_i$, $0 \leq i \leq n-1$

Applying circuit $C_i$ should result in $[i]$, but result $[j]$ is readout error

- $C_i$ is performed $M$ times
- If $[j]$ results $K$ times, then $C_{ij} = K/M$
  $\Rightarrow C_{ij} = \text{Prob( measured value } = j \mid \text{ true } = i)$

$[i]$ : integer value of $|i\rangle$
\[ C_1 \]

|0⟩ – \( X \) – |0⟩

|0⟩ – |0⟩

|0⟩ – |0⟩

|0⟩ – |0⟩

\[ C_{1,0} = 0.03320 \]
\[ C_{1,1} = 0.91406 \]
\[ C_{1,3} = 0.01172 \]
\[ C_{1,5} = 0.00879 \]
\[ C_{1,9} = 0.02539 \]
\[ C_{1,11} = 0.00098 \]
\[ C_{1,13} = 0.00098 \]
\[ C_{1,17} = 0.00391 \]
\[ C_{1,25} = 0.00098 \]

\[ \text{all other } C_{1,i} = 0 \]
Implication of Readout Errors

Correcting readout errors requires additional operations (namely the calibration circuits) to determine the calibration matrix regularly (fortunately not for every execution of the "ideal" algorithm)

Correcting readout errors requires classical post-processing, i.e. applying the calibration matrix to the measured results
Readout Errors: Periodic processing

Preprocessing

Prepare $|0\ldots0\rangle$

Execute $C_s$

Measure $|s\rangle$

Postprocessing

$|s\rangle = |b_0b_1\ldots b_{n-1}\rangle \in \{0,1\}^n$

Generate circuit

$C_s = X^{b_0} \otimes \ldots \otimes X^{b_{n-1}}$

$0 \leq s \leq 2^{n-1}$

$C = (C_s)$

Store $C$
Readout Errors: Postprocessing

State Preparation → Unitary Transformation → Readout

Unfolding:
$t \approx C \cdot m = \hat{t}$
NISQ Analyzer
NISQ Assessment

Error bound $\epsilon$

NISQ Analyzer

Execution possible: Depth? Width?

QC$_1$ or QC$_2$?
Provenance: Definition

- Definition
  - Information describing a process, computation, or data
  - Goals: reproducibility, understandability, quality

- Importance for QC
  - Noisy machines (decoherence, gate infidelity, …)
  - Very different hardware implementations (superconducting, trapped ion, optical, …)
Provenance: Categories

Available gates, topology,…

Used gates, depth,…

SPAM Errors
Provenance Usage

Diagram:
- Algorithm
- Analyzer
- Aggregator
- Provenance Database
- Collector
- Device
- Calibration, Matrix, ...
- Depth, Gates, ...
- Topology, Gates, ...

© Frank Leymann
Hardware Dependent Operations

Set of basic operators is hardware *implementation* dependent

- E.g. continuous-variable (CV) operations in optical quantum computers
  - Squeezing, FockState,... in PennyLane

- E.g. different sets of basic operators implemented by vendors of same category of hardware implementation
  - E.g. U1, U2, U3,… on IBM Q; or Rx(\(\pi/2\)), CZ,… on Rigetti; …

Thus, NISQ compiler must even be aware of implementation of hardware
NISQ Rewriting

Modeling Tool

Translator
- Forest Plugin
- Project Q Plugin
- QISKit Plugin
- ... 

IBMQ_16_Melbourne
(15 Qubits)

QISKit
(IBM)

pyQuil
(Rigetti)

Agave
(8 Qubit)
Rewriting Stages

Data Preparation
Inject circuit that initializes input state

Oracle Expansion
Inject circuit of oracle logic

Gate Mapping
Substitute gates by circuit of vendor supported gates

Metrics
E.g. decoherence time (T1, T2), gate fidelity,…

Execution Readiness

QPU1 \rightarrow \checkmark - Recommendation
QPU2 \rightarrow \times
QPU1 \rightarrow \checkmark
...
QPU_n \rightarrow \times

Recommenation: Deploy on QPU_1, QPU_3

Readout Error Mitigation
Preparation

\[ M^{-1} = \begin{bmatrix} 0.98 & 0.03 \\ 0.02 & 0.97 \end{bmatrix}^{-1} \]

Calibration Matrix
1.02105263 -0.03157895
-0.02105263 1.03157895

© Frank Leymann
NISQ Recommender
Final Remarks
Summary

- NISQ is determined by
  - Decoherence
  - Gate infidelity
  - Readout errors
  - Connectivity

- Data preparation is another problem
- Measurement yet another one

- All these problems can be addressed…
  - …but require additional gates and qbits

- Thus, resources available for proper algorithm is further reduced

- NISQ Analyzer (Rewriter, Recommender,…) will be a tool that helps to determine best QPU to be used for solving a problem based on a given algorithm and given data under constraints like cost, precision,…
The End