# Approximate Quantum State Preparation is Iterated Sparse State Preparation on Linear Neighbor Architectures

Ralph Wang, Akshay Ajagekar, and Fengqi You

Cornell University, Ithaca, NY, USA, 14853

15 September 2023

## Abstract

Quantum algorithms promise exponential speedups in many computational routines. One important subroutine in many such quantum algorithms is quantum state preparation. In this work, we propose a novel, non-variational framework for quantum state preparation by iteratively applying sparse quantum state preparation methods. The proposed framework is flexible with respect to CX gate count constraints, preparation fidelity requirements, and hardware connectivity, without requiring variational tuning. We also give an explicit implementation for the linear nearest neighbor architecture, using CX gates and single-qubit rotations, and compare it against exact state preparation using uniformly controlled gates (UCG) and against variational quantum circuit (VQC) approaches. The proposed method uses 25% fewer CX gates compared to exact state preparation and runs orders of magnitude faster than the VQC approaches.

## 1 Introduction

Quantum algorithms promise exponential speedups in many computation routines, such as quantum Fourier Transform [1] and solving linear systems [2]. Quantum state preparation, the encoding of classical data onto the amplitudes of a quantum computer, is an essential step in such quantum algorithms [3]. Current quantum computers experience significant noise and quantum state preparation often contributes significantly towards the runtime cost of quantum algorithms [3]. Therefore, implementing quantum state preparation efficiently on quantum computers is essential towards useful quantum algorithms on near term quantum computers.

The cost of a quantum circuit is often measured in terms of gate count and circuit depth [4]. Gate count refers to the number of two-qubit gates in the quantum circuit - single-qubit gates are not counted because they are much simpler to implement and induce less noise [4]. Most quantum algorithms use control-X (CX) gates as their two-qubit gate [2, 4, 5]; in this work, we follow suit, and use the number of CX gates as the gate count. Circuit depth refers to the longest subsequence of non-commutative gates in the circuit. The greater the circuit depth, the longer it takes to execute, which leads to more noise from qubit decoherence [6].

The problem of efficient quantum state preparation has received much attention over the years. Shende et al [4] proposed an $O(n2^n)$ algorithm for exact quantum state preparation using $O(2^n)$ CX gates, $O(2^n)$ circuit depth, where $n$ is the number of qubits. Their algorithm starts by rotating the first qubit, then iteratively rotating successive qubits conditioned on preceding qubits, contributing both a general framework of "uniformly-controlled gates." Bergholm et al [7] proposed an optimization on the method from [4], reducing the number of CX gates by a factor of two. Plesch and Bruckner [8] adapted unitary synthesis optimizations from [4] towards quantum state preparation, reducing the gate count by a factor of 1/24. Sun et al proposed a depth-efficient decomposition of uniformly controlled rotations, leading to a quantum state preparation method with a constant factor more CX gates, but with $O(\frac{2^n}{n})$ depth [6]. They also showed how ancilla qubits could be used to compress the circuit depth even further, sacrificing space for time. Zhang et al took this idea one step further, proposing a method that prepares $n$-qubit quantum states using $O(2^n)$ qubits,

but only $O(n)$ depth [9]. All of the proposed algorithms return asymptotically optimal gate counts in a reasonable computation time. Unfortunately, asymptotically optimal is exponential in the number qubits, and such large gate counts cannot be feasibly implemented on near term quantum computers.

Another recent trend in the field has been to study sparse quantum state preparation [10]. Sparse quantum states on $n$ qubits are characterized by having at most $m$ nonzero amplitudes, where $m < 2^n$. The efficient preparation of sparse quantum states find applications in solving linear systems [2], the quantum Byzantine agreement [11], among other quantum algorithms [12]. Many sparse quantum state preparation algorithms have been proposed. Gleinig and Hoefler [13] proposed an algorithm that starts with the target sparse state, then iteratively merges the non-zero amplitudes together until only one non-zero amplitude remains; reversing this process then generates a gate sequence for preparing that non-zero state. By contrast, Malvetti et al [14] proposed a general framework for implementing sparse isometries, which includes quantum state preparation as a subset, although their algorithm takes longer to compute. More recently, sparse state preparation using decision diagrams was proposed to improve merging efficiency, achieving around 30% reduction in CX gate count when $m$ was $O(n^2)$ or larger [10].

These existing algorithms generally assume that CX gates can be applied to any two qubits. However, current state-of-the-art quantum computers, such as IBM's 433-qubit Osprey processor, allow CX gates to only be applied to nearest-neighbor qubits [15]. Each quantum computer has its own physical layout of the qubits; to model these connectivity restrictions, much work has examined adapting quantum algorithms towards linear nearest neighbor architectures, or quantum hardware where the $i$th qubit's nearest neighbors are the $i-1$th and $i+1$th qubit [4, 7, 16, 17]. In particular, [7] showed how their UCG decomposition scheme can be optimized for linear nearest neighbor architectures, however, even with the optimization, the linear nearest neighbor restriction increases the CX gate count by a factor of two.

One popular, hardware-adaptive alternative to exact quantum state preparation has been approximate encoding using variational quantum circuits [18]. A generic, shallow, hardware-efficient quantum circuit is used to approximately prepare quantum states, then a variational optimization procedure is used to adjust the single-qubit gates until the quantum circuit approximately prepares the desired state [19]. This approach has the advantage of being hardware adaptive and often requiring fewer CX gates compared to the exact quantum state preparation methods [20]. However, McClean et al [21] proved that if the quantum circuit structure exhibits no biases towards any particular quantum state, then the gradient with respect to preparation fidelity vanishes exponentially, causing the angle-tuning procedure to take exponential time; the authors dub this the "barren plateau" problem. Local cost functions [20], cost landscape tuning [22], and few-shot approximations [18] have been examined as solutions towards barren plateaus, but the problem remains open.

In this work, we address the problem of approximate, dense state preparation on restricted connectivity hardwares without any variational optimization procedure. Specifically, we propose a novel framework for preparing dense quantum states by iteratively applying sparse quantum state preparation methods. In doing so, we suggest that advances in sparse quantum state preparation can be adapted to improving dense quantum state preparation. To account for potential hardware connectivity restrictions, we analyze our algorithm with respect to linear nearest neighbor connectivity then discuss extensions towards other hardware connectivity graphs. We benchmark our algorithm on randomly sampled dense quantum states, up to 14 qubits, and compare it against the UCG method and two VQC-based methods. By comparing the gate counts and computational runtimes, we demonstrate that our method is able to find quickly find efficient gate sequences for high-fidelity quantum state preparation. The major contributions of this work are summarized as follows:

- A novel, hardware-adaptive, non-variational framework for approximate quantum state preparation is proposed.
- A connection between sparse quantum state preparation and approximate, dense quantum state preparation is established.
- A specific implementation of the proposed framework is described and shown to effectively prepare random quantum states.

In section 2, we define the approximate quantum state preparation problem. In section 3, we describe some sparse state preparation methods then describe our proposed algorithm for dense quantum state preparation using those sparse state preparation methods. In section 4, we compare the proposed algorithm against

exact quantum state preparation and two variational quantum algorithms, comparing their CX gate counts and classical computation times. We discuss the results and conclude in section 5. Some additional technical results are shown in the Appendix.

# 2 Preliminaries

## 2.1 Notation

For this work, qubit indices start at zero, and start from the right. This means $|0010\rangle$ is the result of applying an $X$ gate to qubit 1 of $|0000\rangle$. Single-qubit rotation gates will be written with the rotation angle first, then the qubit index. For example, an RZ gate, angle $\pi/2$, applied to qubit 1, will be written as RZ($\pi$/2, 1). Control-X (CX) gates will be written with the control qubit first. For example, a CX gate applied to qubit 2, with qubit 1 as the control, will be written as CX(1, 2). Unless otherwise specified, $n$ will represent the number of qubits in the system, and $N = 2^n$ will represent the number of amplitudes to be encoded onto those $n$ qubits.

## 2.2 Quantum State Preparation Definition

Quantum state preparation is formally defined as: Given an arbitrary quantum state $|x\rangle$ and a family of quantum gates $G$, return a sequence of quantum gates $g_1, g_2, ..., g_m$ from $G$ such that $g_m g_{m-1} ... g_1 |0\rangle = |x\rangle$ [4].

In this paper, we assume $G$ contains continuously parameterized RX, RY, and RZ gates, as well as the CX gate. This gate set was chosen because these gates are both easy to reason about and easy to compile into the native gate set of existing quantum computers [23].

In this work, we tackle the following version of the approximate quantum state preparation problem: given an arbitrary quantum state $|x\rangle$, return a sequence of quantum gates $g_1, g_2, ..., g_m$ such that

$$g_m g_{m-1} ... g_1 |x\rangle = |y\rangle$$

And

$$|\langle y|0\rangle|^2 \geq 1 - \epsilon$$

Where $\epsilon$ represents a small error. The quantity $1 - \epsilon$ is also known as the fidelity.

This version of approximate quantum state preparation can be applied towards approximately preparing arbitrary quantum states. If for some state $|x\rangle$, a sequence of gates $g_1$, $g_2$, ..., $g_m$ can be found for transforming $|x\rangle$ to $|y\rangle$, a state close to $|0\rangle$, then inverting each gate and reversing the sequence generates a sequence of gates for approximately preparing $|x\rangle$ starting from $|0\rangle$. If the gate sequence $g_m^\dagger$, $g_{m-1}^\dagger$, ..., $g_1^\dagger$ was applied to $|0\rangle$ to achieve the state $|x'\rangle$, then

$$|\langle x|x'\rangle|^2 = |\langle x| (g_1^\dagger g_2^\dagger ... g_m^\dagger |0\rangle)|^2$$

$$= |\langle y|0\rangle|^2 \geq 1 - \epsilon$$

Thus showing that the prepared state has a fidelity of $1 - \epsilon$.

# 3 Proposed Algorithm, Formal Description

In this work, we propose an approximate quantum state algorithm, which we call "iterative sparse approximation" (ISA). Let the target state be $|x\rangle$, and let $l$ be a list of quantum gates. Let the *current state*, $|c\rangle$, be the state achieved by applying each gate in $l$ to $|x\rangle$. As the algorithm proceeds, gates will be added to $l$. When the algorithm terminates, $l$ will be returned. The current state $|c\rangle$ starts out as $|x\rangle$ but becomes a state close to $|0\rangle$ by the end of the algorithm. Let $c_i$ denote the amplitude at basis gate $|i\rangle$ in the current state, that is,

$$c_i = \langle i|c\rangle$$

The $|c\rangle$ and $c_i$ values are updated as gates are added to $l$.

The algorithm starts by using RZ and RY gates to bring the current state closer to $|0\rangle$ (Refinement without CX gates), then iterates sparse approximations of the current state to bring the current state as close to $|0\rangle$ as needed (Refinement by ISA). The algorithm can be described by the pseudo-code:

**Algorithm 1** Algorithm 1: ISA framework

```
Initialize l to the empty list
Initialize |c⟩ = |x⟩
//Refinement by RZ-RY
for i in [0 ... n − 1] do
    Compute θ_i and φ_i
    l += RZ(i, φ_i)
    l += RY(i, θ_i)
    Update |c⟩ = RY(i, θ_i)RZ(i, φ_i) |c⟩
end for
//Refinement by ISA
while |⟨0|c⟩|^2 < 1 − ε do                ▷ check termination condition
    |c'⟩ = Sparse approximation(|c⟩)
    s = Sparse prepare(|c'⟩)
    s' = Adapt(g, |c⟩)
    for g in s' do
        l += g
        Update |c⟩ = g |c⟩
    end for
end while
return l
```

In the remainder of this section, we describe the details of the algorithm. First, we show the sparse quantum state preparation methods to be adapted to our algorithm. Next, we describe the details of refinement by RZ-RY. Then, we elaborate on the method for selecting a sparse approximation in refinement by ISA, before finishing with the method for adapting sparse quantum state preparation for dense quantum states.

## 3.1 Sparse Quantum State Preparation

This work relies on the efficient preparation of two special types of sparse quantum states. We enumerate these special states here, along with their efficient preparation method. Our preparation methods presented here are inspired by the methods presented in [14].

The first type of special sparse quantum states, which we denote "Type 1 sparse states," contains exactly two non-zero amplitudes. To prepare such states, first use a sequence of $RY(\pi)$ gates to move one of the amplitudes to the $|0\rangle$ basis vector. Next, a sequence of CX gates is used to move the other amplitude to a basis vector $|a\rangle$ such that $a$ is a power of 2. Finally, an RZ and an RY gate can be used to merge the two amplitudes together, transforming the starting (target) state to $|0\rangle$. For example, if

$$|x\rangle = \frac{\sqrt{2}}{2}|010\rangle + \frac{\sqrt{2}}{2}i|101\rangle$$

First, $\mathrm{RY}(\pi, 1)$ is applied to move $\frac{\sqrt{2}}{2}|010\rangle$ to the $|000\rangle$ position:

$$\mathrm{RY}(\pi, 1)|x\rangle = \frac{\sqrt{2}}{2}|000\rangle - \frac{\sqrt{2}}{2}i|111\rangle := |x_1\rangle$$

Next, CX gates are applied to move the other amplitude to a basis vector $|a\rangle$ such that $a$ has exactly one 1-bit in its binary representation. In this case, we arbitrarily choose $a = 100$. Then, CX(1, 0) and CX(2, 1) need to be applied in sequence to move $|111\rangle$ to $|100\rangle$.

$$\mathrm{CX}(2,1)\mathrm{CX}(1,0)|x_1\rangle = \frac{\sqrt{2}}{2}|000\rangle - \frac{\sqrt{2}}{2}i|100\rangle := |x_2\rangle$$

Next, an RZ gate is used to zero out the complex phase on $|100\rangle$:

$$\mathrm{RZ}(\pi/2, 2)\,|x_2\rangle = \frac{\sqrt{2}}{2}\,|000\rangle + \frac{\sqrt{2}}{2}\,|100\rangle := |x_3\rangle \text{ (ignoring global phase)}$$

Finally, an RY gate is used to merge the amplitudes together:

$$\mathrm{RY}(-\pi/2, 2)\,|x_3\rangle = |000\rangle$$

The second type of special sparse quantum state, denoted "Type 2 sparse states," are $n$-qubit states that can be written in the form:

$$|x\rangle = (|a_p\rangle \otimes |x_1\rangle + |0_p\rangle \otimes |x_2\rangle) \otimes |0_q\rangle \tag{1}$$

Where $|x_1\rangle$ and $|x_2\rangle$ are arbitrary, non-normalized two-qubit states, $|a_p\rangle$ is a nonzero $p$-bit basis state, $|0_p\rangle$ is a $p$-bit basis state, $|0_q\rangle$ is a $q$-bit basis state ($q$ may be zero), and $p + q + 2 = n$. Type 2 sparse states are those that can be written as the sum of two two-qubit states and contain at most eight non-zero amplitudes.

To efficiently prepare such states, CX gates are first used to move $|a\rangle$ to $|1\rangle$. The resultant quantum state will be a dense, three-qubit state, which can be prepared exactly using three CX gates. Optimal, exact three-qubit state preparation is well known in the literature [24]; we describe our implementation in the Appendix.

## 3.2 Refinement by RZ-RY

First, the current state's maximal amplitude basis state, $|b\rangle$, is identified. While there is a qubit that does not yet have any gates applied to it, do:

1. Collect the set of qubit indices that do not yet have any gates applied to them, call this set $I$.
2. For each $i$ in $I$, compute $b_i$ as the result of flipping the $i$th bit in $b$.
3. Select $i$ such that $c_{b_i}$ is maximal.
4. Compute $\theta_z = phase(c_b) - phase(c_{b_i})$. If $b_i < b$, flip the sign of $\theta_z$.
5. Compute $\theta_y = arctan(|\frac{c_b}{c_{b_i}}|)$. If $b_i < b$, flip the sign of $\theta_y$.
6. Add RZ($\theta_z$, $i$) and RY($\theta_y$, $i$) to $l$, in that order. Update $|c\rangle$, and set $b \leftarrow min(b, b_i)$

Each time step 4 of the loop is run, the $i^*$th bit in $b$ is set to 0. Thus, at the end of this procedure, $b = 0$, and the amplitude at $|0\rangle$ of the current state is at least as large as the largest amplitude in $|x\rangle$.

## 3.3 Refinement by ISA: Selecting a sparse approximation

To select a sparse approximation, each possible sparse approximation is enumerated. Then, the *fidelity increase ratio* is computed for each candidate sparse approximation. Finally, the sparse approximation with the largest fidelity increase ratio is selected.

### 3.3.1 Type 1 sparse approximations

During refinement by RZ-RY, the largest amplitude in the $|x\rangle$ was moved to $|0\rangle$ in $|c\rangle$. Thus, if $|c\rangle$ is to be approximated by a Type 1 sparse state, one of the non-zero amplitudes is chosen to be $|0\rangle$. The other amplitude can be chosen freely. Thus, we index Type 1 sparse approximations by the other non-zero amplitude index:

$$|x_1(i)\rangle = c_0\,|0\rangle + c_i\,|i\rangle$$

The subscript 1 refers to Type 1 sparse approximation; the $i$ refers to the index of the non-zero amplitude. In general, these sparse approximations are non-normalized states. The corresponding fidelity increase ratio is:

$$\text{fidelity increase ratio}(|x_1(i)\rangle) = \frac{|c_i|^2}{\alpha + \mathrm{CXdist}(i)}$$

Where CXdist($i$) is the minimum number of CX gates required to transform $|i\rangle$ to $|i'\rangle$ such that $i'$ is a power of 2. This is also the number of CX gates required to prepare $|x_1(i)\rangle$, as shown in section 4.1. The algorithm for computing CXdist($i$) will be given in the Appendix.

The term $\alpha$ is a regularization term to prevent division by zero. For our implementation, we set $\alpha = 1$.

#### 3.3.2 Type 2 sparse approximations

Each Type 2 sparse state contains eight non-zero amplitudes; the indices of those non-zero amplitudes are uniquely determined by $p$ and $a_p$ eqn. (1). As with Type 1 sparse approximation, we index each candidate Type 2 sparse approximation by its non-zero amplitudes:

$$|x_2(a, a_p)\rangle = ((|a_p\rangle \langle a_p| + |0_p\rangle \langle 0_p|) \otimes I \otimes I \otimes |0_q\rangle \langle 0_q|) |c\rangle$$

In effect, $|x_2(a, a_p)\rangle$ is $|c\rangle$ but with all but the specified eight amplitudes set to zero. Like Type 1 sparse approximations, the Type 2 sparse approximations are generally non-normalized states.

For some non-normalized two-qubit states $|\psi(a, a_p)_1\rangle$, $|\psi(a, a_p)_2\rangle$. Then, the fidelity increase ratio can be computed as:

$$\text{fidelity increase ratio}(a,\, a_p) = \frac{|\langle x_2(a, a_p)|x_2(a, a_p)\rangle|^2 - |c_0|^2}{\alpha + \text{CXdist}_1(a_p) + 3}$$

Where $\text{CXdist}_1(a)$ is the number of CX gates required to transform $|a\rangle$ to $|1\rangle$. The plus three in the denominator accounts for the three CX gates required to perform the three-qubit quantum state preparation base case, thus, $\text{CXdist}_1(a_p) + 3$ is the number of CX gates required to prepare $|x_2(a, a_p)\rangle$. The algorithm for computing $\text{CXdist}_1$ will be given in the Appendix.

To choose a sparse approximation for the current state, all possible Type 1 and Type 2 sparse approximations are considered and the approximation with the largest fidelity increase ratio is selected.

### 3.4 ISA: Adapting sparse approximation towards state preparation

Once a sparse approximation $|x'\rangle$ is selected, the corresponding state preparation method is applied to $|c\rangle$ to bring it closer to $|0\rangle$.

#### 3.4.1 Type 1 sparse approximation

If a Type 1 sparse approximation is selected, then the non-normalized sparse approximation takes the form $|c\rangle \approx |c'\rangle = c_0 |0\rangle + c_b |b\rangle$ for some $b$. The corresponding state preparation method moves $|b\rangle$ to a power of 2, then merges it with $|0\rangle$ using an RZ and an RY gate. This is adapted to $|c\rangle$ by the following procedure:

While $b$ is not a power of 2, do:

1. Enumerate all $b'$ such that $b' \neq b$ and there exists some CX gate that transforms $|b\rangle$ to $|b'\rangle$. Let the set of all such $b'$ be denoted $B$.

2. For each $b'$ in $B$, calculate the move increase ratio as:

$$\text{move increase ratio}(b') = \frac{|c_b|^2 + |c_{b'}|^2}{1 + min(\text{CXdist}(b), \text{CXdist}(b'))}$$

3. Select $b^*$ from $B$ such that move increase ratio($b^*$) is maximal.

4. If $\text{CXdist}(b) < \text{CXdist}(b^*)$, then merge the amplitudes at $|b\rangle$ and $|b'\rangle$ into $|b\rangle$ without moving the amplitude at $|0\rangle$. Otherwise, merge the amplitudes into $|b^*\rangle$ without moving the amplitude at $|0\rangle$ and assign $b \leftarrow b^*$.

After this loop terminates, merge the amplitudes at $|b\rangle$ and $|0\rangle$ into $|0\rangle$.

To merge the amplitudes at $|a\rangle$ and $|b\rangle$ into $|b\rangle$, without moving the amplitude at $|0\rangle$, use the following procedure:

1. For this merging to be possible, $a$ and $b$ must differ by exactly one bit in their binary representation. Let this difference be at position $i$. Also, let $c_a$ be the amplitude of the current state at $|a\rangle$ and let $c_b$ be the amplitude at $|b\rangle$. Also, $a$ and $b$ must both have a 1 adjacent to position $i$ in their binary representation, let this position be $j$.

2. Compute $\theta_z = phase(c_b) - phase(c_a)$. If $a < b$, flip the sign of $\theta_z$.

3. Compute $\theta_y = arctan(|\frac{c_b}{c_a}|)$. If $a < b$, flip the sign of $\theta_y$.

4. Add RZ($\theta_z$, $i$), RY($\theta_y$, $i$), CX($j$, $i$), and RY($-\theta_y$, $i$), in that order, to $l$ and update $|c\rangle$.

This procedure usually changes the complex phase of $c_0$, but this is inconsequential towards the algorithm.

To merge the amplitude at $|b\rangle$ into $|0\rangle$, the method from refinement by RZ-RY can be applied.

### 3.5 Implementing a Type 2 sparse approximation

If a Type 2 sparse approximation is selected, then the sparse approximation takes the form:

$$|c\rangle \approx |c'\rangle = (|a_p\rangle \otimes |x_a\rangle + |0_p\rangle \otimes |x_0\rangle) \otimes |0_q\rangle$$

Where $|x_a\rangle$ and $|x_0\rangle$ are non-normalized two-qubit states. Define $|x_i\rangle$ as:

$$|x_i\rangle = (|i\rangle \langle i| \otimes I \otimes I \otimes |0_q\rangle \langle 0_q|) |c\rangle$$

The sparse state preparation method for Type 2 sparse approximations moves $|a\rangle$ to $|1\rangle$, then use three-qubit state preparation to merge $|1\rangle \otimes |x_1\rangle$ into $|0\rangle \otimes |x_0\rangle$. This method can be adapted to $|c\rangle$ by the following procedure.

1. Enumerate all $a'$ such that $a \neq a'$ and $|a\rangle$ can be transformed to $|a'\rangle$ by applying a CX gate; let the set of all such $a'$ be denoted $A$.

2. For each $a'$ in $A$, compute its move increase ratio as:

$$\text{move increase ratio}(a') = \frac{\frac{1}{2}(\langle x_{a'}|x_{a'}\rangle) + \frac{1}{2}\sqrt{(\langle x_{a'}|x_{a'}\rangle - \langle x_a|x_a\rangle)^2 + 4|\langle x'_a|x_a\rangle|^2}}{3 + min(\text{CXdist}_1(a), \text{CXdist}_1(a'))}$$

   The numerator represents the magnitude of $|x'_a\rangle$ after approximately merging $|a\rangle$ into $|a'\rangle$; the denominator represents one CX gate used to perform this merging plus the number of CX gates remaining in the implementation of this sparse approximation.

3. Select $a^*$ to be the element of $A$ with the largest fidelity increase ratio

4. If $\text{CXdist}_1(a) < \text{CXdist}_1(a^*)$, then merge $|a^*\rangle \otimes |x_{a^*}\rangle$ into $|a\rangle \otimes |x_a\rangle$ while leaving $|0\rangle \otimes |x_0\rangle$ unchanged. Otherwise, merge $|a\rangle \otimes |x_a\rangle$ into $|a^*\rangle \otimes |x_{a*}\rangle$ and assign $a \leftarrow a^*$.

After the loop terminates, $a = 1$, and the three-qubit state preparation procedure can be used to merge $|a\rangle$ into $|0\rangle$.

Merging $|a\rangle \otimes |x_a\rangle$ into $|b\rangle \otimes |x_b\rangle$ while leaving $|0\rangle \otimes |x_0\rangle$ unchanged usually cannot be done exactly using three or fewer CX gates. Therefore, in our implementation, we perform this merging only approximately. In addition, this can be done only if $|a\rangle$ can be transformed to $|b\rangle$ using some CX gate, CX($i$, $j$). Then, approximate merging is implemented by the following procedure:

1. If $a < b$, let $|v_0\rangle = |x_a\rangle$ and $|v_1\rangle = |x_b\rangle$. Otherwise, let $|v_1\rangle = |x_a\rangle$ and $|v_0\rangle = |x_b\rangle$.

2. Compute $\theta_z = -phase(\langle v_0|v_1\rangle)$.

3. Compute $\theta_y = \frac{1}{2}\left(-\pi - arccos\left(\frac{\langle v_0|v_0\rangle - \langle v_1|v_1\rangle}{2\sqrt{(\langle v_0|v_0\rangle - \langle v_1|v_1\rangle)^2 + |\langle v_0|v_1\rangle|^2}}\right)\right)$

4. Add RZ($\theta_z$, $j$), RY($\theta_y$, $j$), CX($i$, $j$), and RY($-\theta_y$, $j$) to $l$ and update $|c\rangle$.

# 4 Numerical Experiments: CX count and computation time on random dense states

## 4.1 Method

We compare our ISA method against the exact state preparation method using uniformly controlled gates (UCG) [7], alternating ansatz VQC [5], and ADAPT-VQE [25] methods. All experiments were performed assuming the linear nearest neighbors architecture; for the proposed method and variational methods, the target state preparation fidelity was set to 0.95. For the variational methods, simple gradient descent was used, with $1 - fidelity$ as the cost function, and a constant learning rate of 0.1. For ADAPT-VQE, initialized the circuit to contain a layer of RY rotations, followed by a layer of RZ rotations. For the operator pool, we used the following circuit block, translated across different possible positions:

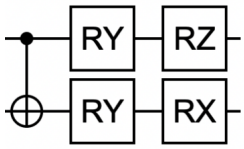


Figure 1: *The circuit block used in ADAPT-VQE for state preparation. The operator pool is generated by translating this block across different pairs of qubits.*

In addition, we modified the ADAPT-VQE procedure. Instead of comparing gradient sizes to choose an operator, we performed 30 gradient descent steps for each candidate operator and selected the candidate that gave the best improvement. Then, after adding that operator, the entire gate sequence was trained for 200 additional steps.

The implementation of the UCG method was taken from the Python package *qclib* [26]. All other methods were implemented from scratch in C++.

In our experiments, we tested the number of CX gates used and the computation time as a function of number of qubits in the target state. We tested qubit numbers from 5 to 14, inclusive. For each qubit number, 100 quantum states were uniformly, randomly sampled.

For the runtime experiments, the number of CX gates in the alternating ansatz VQC was set to $2^{n-1}$. This number was specifically chosen because each CX gate attaches four free parameters, therefore, approximately $2^{n-1}$ CX gates are necessary for the number of circuit parameters to exceed the number of free parameters in an $n$-qubit quantum state. Under these conditions, the VQC should be able to prepare any arbitrary $n$-qubit quantum state with high fidelity.

To determine the number of training iterations needed for alternating ansatz VQC, we trained VQC's with varying numbers of layers, and report the CX gate count corresponding to the minimum number of layers needed to achieve an average fidelity of 0.95. However, this minimum CX count also depends on the number of gradient descent steps used to optimize the angles: more layers requires less training. To ensure a sufficiently large but also fair number of training iterations for each $n$, we trained an alternating ansatz with $\frac{1}{2}2^n$ CX gates on the first three random quantum states and recorded the number of training iterations required to achieve 0.95 fidelity on all three states. Then, the number of training iterations was set to the sum of those numbers multiplied by two. This ensured that the number of training iterations was approximately six times as many as actually necessary for that specific number of qubits, preventing insufficient training from marring the results.

Finally, CX gate count experiments were not run for the uniformly controlled gates method. This is because the number of gates for this exact method depends on the transpiler, sometimes finding good optimizations for the linear nearest neighbor architecture, sometimes failing to. However, using the exact nature of this algorithm, it was determined that this method uses $2 \times 2^n + 2n - 19$ CX gates to prepare an $n$-qubit quantum state (proof given in the Appendix).

## 4.2 Results

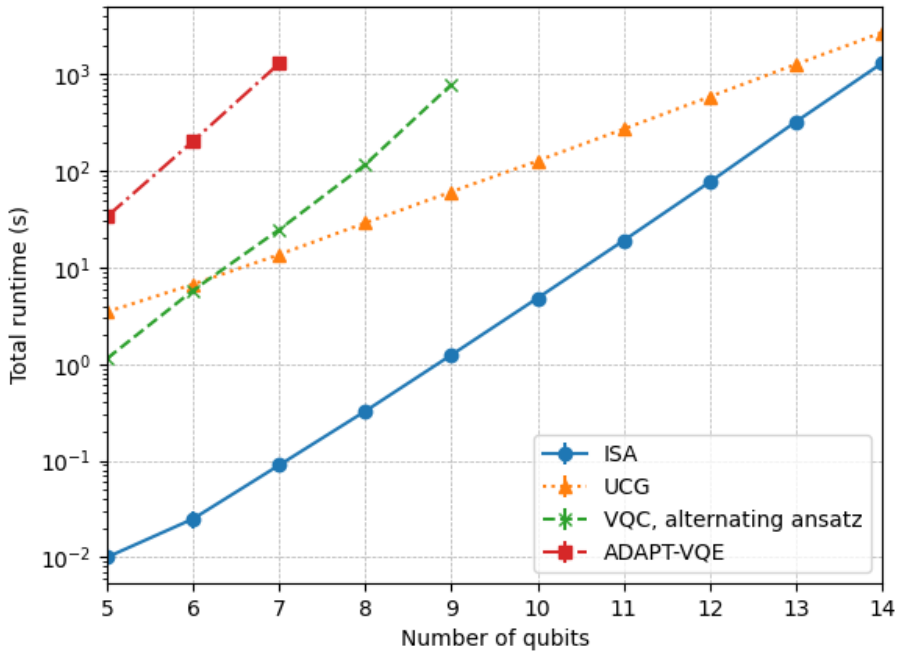


Figure 2: *Total runtime versus number of qubits for each quantum state preparation method, plotted on a linear-log scale. Only the ISA and UCG methods were able to prepare the quantum states for $n$ up to 14; the alternating ansatz method timed out after $n = 9$ and the ADAPT-VQE method timed out after $n = 7$. In addition, while the ISA method starts out much faster than the UCG method, the UCG curve has a lower slope than the ISA method, indicating that for larger $n$, UCG is probably faster than ISA.*

Figure 2 shows the runtime versus number of qubits curve for each method. For all numbers of qubits tested, the iterated sparse approximation method had the lowest runtime, however, the ISA method's runtime increases faster with qubit number compared to the UCG method. Thus, the UCG method is expected to be faster than ISA for more than 14 qubits; both methods were orders of magnitude faster than the variational methods. This reflects the intense computational cost incurred by rotation angle optimization procedure.

Table 1 shows the average number of CX gates used by each method as a function of the number of qubits. The UCG method uses the most CX gates; the ISA method uses somewhat fewer CX gates, while the variational methods use the fewest CX gates. The variational methods far outperform the non-variational methods in this regard, demonstrating the effectiveness of the variational optimization procedure. That said, the ISA method is shown to outperform the UCG method without any variational angle tuning procedure, indicating that iterative sparse approximation is able to directly find good quantum state preparation gate sequences.

The number of free parameters in the quantum circuit is proportional to the number of CX gates; the number of free parameters needed is proportional to $N = 2^n$, the number of complex amplitudes in the target state. Therefore, we hypothesized that the number of CX gates would be approximately proportional to $N$ and try to determine the constant factor by graphing CX count divided by $N$ as a function of the number of qubits. If the CX count is indeed proportional to $N$, then the graph should approach a horizontal line for large $n$, with its asymptote being the constant factor.

| Number of qubits | ISA | UCG | ADAPT-VQE | VQC |
|---|---|---|---|---|
| 5 | 24.08 | 55 | 13.67 | 14 |
| 6 | 60.98 | 121 | 25.97 | 28 |
| 7 | 143.46 | 251 | 47.96 | 48 |
| 8 | 319.02 | 509 | 88.76 | 91 |
| 9 | 689.32 | 1023 | 164.22 | 180 |
| 10 | 1439.6 | 2049 | TIMEOUT | TIMEOUT |
| 11 | 2952.66 | 4099 | | |
| 12 | 5991.51 | 8197 | | |
| 13 | 12123.3 | 16391 | | |
| 14 | 24538 | 32777 | | |

Table 1: *Average number of CX gates used to prepare 100 randomly generated quantum states versus number of qubits. Therefore, the values in the ISA and ADAPT-VQE column show two decimal digits. By contrast, the values for the UCG column were calculated, and are therefore whole numbers. In addition, the values for VQC, alternating ansatz were calculated by finding the minimum number of CX gates corresponding to an average fidelity of 0.95 - these are also whole numbers*

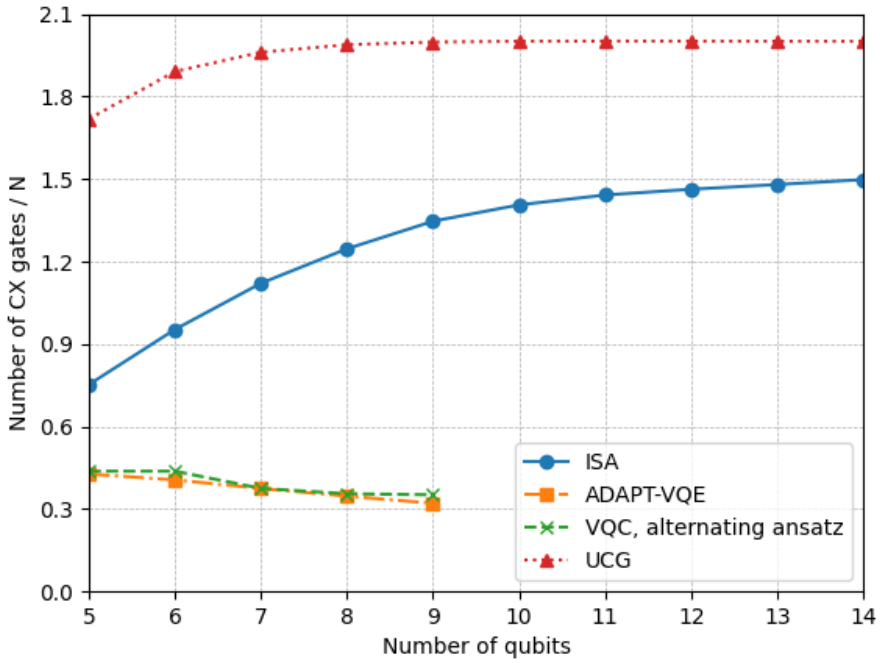


Figure 3: *Average number of CX gates divided by N versus number of qubits for each quantum state preparation method. The curve for VQC and ADAPT-VQE methods are truncated because these methods timed out for larger numbers of qubits. The curve for UCG is a nearly horizontal line at constant factor 2, reflecting the CX count formula for the UCG method. The curve for ISA starts out quite low but increases to around 1.5. The VQC and ADAPT-VQE curves stay quite low, approaching 0.3, but the curves are truncated due to timeout.*

Figure 3 shows the average number of CX gates divided by N as a function of qubit number for each method. For the ISA method, the curve increases for $n < 10$, then stabilizes to around 1.5. Thus, we conclude that the ISA method will use approximately $1.5N$ CX gates for randomly sampled quantum states. By contrast, the UCG method is shown to use $2N + 2n - 19$, which is $2N$ to leading order. Thus, the

ISA method is shown to use approximately 25% fewer CX gates compared to the UCG method. However, both variational methods stabilize to around 0.3, which is about 5 times better than ISA, demonstrating the effectiveness of the variational tuning procedure.

# 5 Discussion and Conclusion

The ISA algorithm presented has been shown to use fewer CX gates than exact state preparation methods without incurring the variational optimization costs associated with VQC algorithms. However, the VQC methods are able to prepare quantum states using much fewer CX gates compared to the ISA method, and the UCG method is expected to be faster than ISA for quantum states on more than 14 qubits. This trend is expected: the less computation time a method uses, the lower the quality of its output.

In practice, every CX gate applied introduces hardware noise into the computation. Therefore, the design of quantum circuits will require a careful balance between using enough CX gates to approximate the desired computation, while not using so many CX gates as to corrupt the computation with noise. The ISA algorithm lends itself to a simple method for finding this balance: if too many CX gates are present in the state preparation algorithm, simply delete the gates corresponding to the last iterative sparse approximation; if insufficient CX gates are present, perform more sparse approximation iterations. This works because each iteration of sparse approximation monotonically increases the state preparation fidelity, and each additional iteration provides diminishing returns on fidelity increase. By contrast, applying such a procedure towards simple VQC would require re-training the quantum circuit depending on the number of ansatz layers used. For large numbers of qubits, this would incur significantly more computational costs.

In addition, different quantum hardwares have different qubit connectivities. The only hardware-dependent steps in the ISA algorithm are enumerating the list of available CX gates and computing CXdist, $\text{CXdist}_1$. The former is derived directly from the hardware connectivity; the latter can be efficiently pre-computed from the list of available CX gates using the worklist algorithm.

However, the ISA method does face several limitations. Most importantly, the ISA methods uses four or five times as many CX gates as the VQC methods. This may will cause ISA-generated gate sequences to induce much more noise on near term quantum hardware compared to VQC-based methods. This problem may eventually be mitigated by future work discovering alternative, easy-to-prepare classes of quantum states for approximating dense quantum states, then adapting those discoveries towards improving the ISA algorithm. In addition, it may be possible to incorporate some variational training alongside ISA. However, the cost function, training schedule, gradient conditioning, and other things would need to be carefully designed to ensure such a hybrid method doesn't become a worse version of ADAPT-VQE or pure VQC.

Another limitation of the ISA method is that it cannot prepare quantum states that require extremely high fidelity. This is because, at each iteration, the ISA method treats the dense target state as a sparse approximate state. Therefore, some percentage of the fidelity will be lost on every iteration of the sparse approximation, and chasing down these lost details will require many, many more iterations of ISA, leading to extremely long gate sequence. That said, many applications of quantum state preparation, such as machine learning and probability distribution function sampling, require only approximate quantum state preparation.

To conclude, we present a non-variational framework for approximate quantum state preparation by iteratively applying sparse quantum state preparation methods. The non-variational nature of our framework resulted in orders of magnitude speedup compared to variational approaches. In addition, the ISA method was able to prepare quantum states with 95% fidelity, using 25% fewer CX gates compared to the UCG exact method on linear nearest neighbor architectures, demonstrating its ability to find good state preparation sequences. Future work may investigate closing the gap in CX gate count between the ISA method and VQC-based methods, perhaps by adapting more advanced sparse state preparation techniques or by selectively integrating variational tuning procedures. Still, the advantages of our method have been demonstrated.

# 6 Appendix

## 6.1 Optimal, exact three-qubit state preparation

Optimal, exact three-qubit quantum state preparation has been studied in the literature [24]; our specific implementation requires three steps, called "D1", "D2", and "SP2."

Let the target state be:

$$|x\rangle = \sum_{i,j,k\in 0,1} a_{ijk} |ijk\rangle$$

Step D1 applies a single-qubit rotation, then CX(0, 1), then another single-qubit rotation to $|x\rangle$, resulting in the state

$$|y\rangle = \sum_{i,j,k\in 0,1} b_{ijk} |ijk\rangle$$

Such that

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} \text{ and } \frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}$$

Next, step D2 uses a single-qubit rotation, CX(1, 2), and another single-qubit rotation to transform $|y\rangle$ into a two-qubit state:

$$|z\rangle = \sum_{j,k\in 0,1} c_{jk} |0jk\rangle$$

.

Finally, step SP2 implements two-qubit state preparation on this state using one CX gate and several single-qubit rotations. We describe each of these steps in detail, starting with SP2 and D2. Then, we briefly discuss the decomposition of uniformly controlled single-qubit rotations before concluding with a description of D1.

### 6.1.1 SP2

Starting from the state

$$|z\rangle = \sum_{j,k\in 0,1} c_{0jk} |0jk\rangle$$

We apply an RZ and RY gate to qubit 0 to merge $c_{001} |001\rangle$ into $c_{000} |000\rangle$ using the method from section 4.4.1 of the main text, leading to the state:

$$|z'\rangle = c'_{000} |000\rangle + c'_{010} |010\rangle + c'_{011} |011\rangle$$

Next, an RZ, RY, CX, and RY gate sequence is used to merge $c'_{011} |011\rangle$ into $c'_{010} |010\rangle$ without moving $c'_{000} |000\rangle$, using the method described in section 4.4.1 of the main text. This results in a one-qubit state that can be prepared using a single qubit rotation gate.

### 6.1.2 D2

Starting from the state

$$|y\rangle = \sum_{i,j,k\in\{0,1\}} b_{ijk} |ijk\rangle$$

We apply an RZ and RY gate to qubit 2 to merge $b_{100} |100\rangle$ into $b_{000} |000\rangle$, using the same method as in SP2. The pre-condition:

$$\frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}$$

Implies that this procedure also merges $b_{101} |101\rangle$ into $b_{001} |001\rangle$. This results in the state:

$$|y'\rangle = \sum_{i,j,k\in\{0,1\}} b'_{ijk} |ijk\rangle \qquad\qquad b'_{100}, b'_{101} = 0$$

In addition, the RZ and RY gate were applied to qubit 2, therefore, the equality

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}}$$

Is preserved after this transformation:

$$\frac{b'_{110}}{b'_{010}} = \frac{b'_{111}}{b'_{011}}$$

Next, an RZ, RY, CX, RY sequence is used to merge $b'_{110}|110\rangle$ into $b'_{010}|010\rangle$ without moving $b'_{000}|000\rangle$, using the same method as in SP2. The equality

$$\frac{b'_{110}}{b'_{010}} = \frac{b'_{111}}{b'_{011}}$$

Ensures that $b'_{111}|111\rangle$ is also merged into $b'_{011}|011\rangle$ by this process. The single qubit rotations are applied to qubit 2 and the CX gate is applied to qubits 1 and 2. Therefore, this operation acts independently of qubit 0. Therefore, $b'_{001}|001\rangle$ is unmoved.

The previous step zeroed out the $|100\rangle$ and $|101\rangle$ basis vectors; this step zeroed out the $|110\rangle$ and $|111\rangle$ basis vector.s Therefore, the result of these transformations is a two-qubit state:

$$|z\rangle = \sum_{j,k\in\{0,1\}} c_{jk}|0jk\rangle$$

### 6.1.3 Decomposing uniformly single controlled single-qubit rotations

In this section, we describe the set of uniformly controlled single-qubit rotations with one control qubit that can be decomposed into one CX gate and single-qubit rotations. In addition, we give a procedure for their decomposition. These constructions will be necessary for the construction of step D1 for optimal, exact three-qubit state preparation.

For simplicity, let the control qubit be qubit 1, let the target qubit be qubit 0. Suppose the desired gate applies $a$ if the control qubit is 0, applies $b$ otherwise. Then the desired gate has matrix form:

$$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

We wish to decompose this transformation into a quantum circuit of the form:

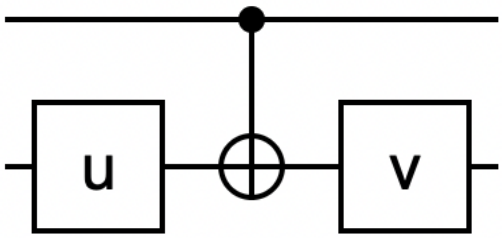


Figure 4: *Target decomposition of the uniformly controlled, single-qubit rotation with one control qubit*

These gates, together, take the matrix form:

$$\begin{bmatrix} v & 0 \\ 0 & v \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} \begin{bmatrix} u & 0 \\ 0 & u \end{bmatrix}$$

This requires some $u$ and $v$ to exist, such that $a = vu$ and $b = vXu$. Then

$$ab^\dagger = (vu)(vXu)^\dagger = vXv^\dagger$$

Which implies that $ab^\dagger$ must have the same eigenvalues, $\pm 1$, as $X$. This condition is both necessary and sufficient for the uniformly controlled rotation to be decomposable into the quantum circuit shown. Indeed, if $ab^\dagger$ has the same eigenvalues as $X$, then we can compute $v$ by diagonalization, then compute $u = v^\dagger a$. Finally, $u$ and $v$ can be decomposed to RY and RZ gates using ZYZ decomposition [4].

### 6.1.4 Implementing D1

Step D1 is implemented using a uniformly controlled single-qubit rotation, with qubit 0 as the control, qubit 1 as the target. To implement D1, we construct single-qubit operations $a$ and $b$ such that applying $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$ leads to the state

$$|y\rangle = \sum_{i,j,k \in \{0,1\}} b_{ijk} |ijk\rangle$$

Such that

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} \text{ and } \frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}$$

In addition, we perform this construction such that $ab^\dagger$ has eigenvalues $\pm 1$ to ensure the transformation can be implemented using only one CX gate.

We parameterize $a$ and $b$ as:

$$a = e^{i\phi_g} \begin{bmatrix} cos(\theta_a) & -sin(\theta_a)e^{i\phi_a} \\ sin(\theta_a)e^{i\alpha_a} & cos(\theta_a)e^{i(\alpha_a+\phi_a)} \end{bmatrix}$$

$$b = \begin{bmatrix} cos(\theta_b) & -sin(\theta_b)e^{i\phi_b} \\ sin(\theta_b)e^{i\alpha_b} & cos(\theta_b)e^{i(\alpha_b+\phi_b)} \end{bmatrix}$$

Then, we can express the $b_{ijk}$ coefficients of $|y\rangle$ in terms of $\phi_g$, $\theta_a$, $\alpha_a$, $\phi_a$, $\theta_b$, $\alpha_b$, $\phi_b$, and the $a_{ijk}$ coefficients of $|x\rangle$:

$$b_{000} = e^{i\phi_g}(cos(\theta_a)a_{000} + sin(\theta_a)e^{i\phi_a}a_{010})$$
$$b_{010} = e^{i(\alpha_a+\phi_g)}(-sin(\theta_a)a_{000} + cos(\theta_a)e^{i\phi_a}a_{010})$$
$$b_{100} = e^{i\phi_g}(cos(\theta_a)a_{100} + sin(\theta_a)e^{i\phi_a}a_{110})$$
$$b_{110} = e^{i(\alpha_a+\phi_g)}(-sin(\theta_a)a_{100} + cos(\theta_a)e^{i\phi_a}a_{110})$$
$$b_{001} = cos(\theta_b)a_{001} + sin(\theta_b)e^{i\phi_b}a_{011}$$
$$b_{011} = e^{i\alpha_b}(-sin(\theta_b)a_{001} + cos(\theta_b)e^{i\phi_b}a_{011})$$
$$b_{101} = cos(\theta_b)a_{101} + sin(\theta_b)e^{i\phi_b}a_{111}$$
$$b_{111} = e^{i\alpha_b}(-sin(\theta_b)a_{101} + cos(\theta_b)e^{i\phi_b}a_{111})$$

Then the constraints

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} and \frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}$$

Can be written as:

$$\frac{e^{i(\alpha_a+\phi_g)}(-sin(\theta_a)a_{100} + cos(\theta_a)e^{i\phi_a}a_{110})}{e^{i(\alpha_a+\phi_g)}(-sin(\theta_a)a_{000} + cos(\theta_a)e^{i\phi_a}a_{010})} = \frac{e^{i\alpha_b}(-sin(\theta_b)a_{101} + cos(\theta_b)e^{i\phi_b}a_{111})}{e^{i\alpha_b}(-sin(\theta_b)a_{001} + cos(\theta_b)e^{i\phi_b}a_{011})}$$
$$\frac{e^{i\phi_g}(cos(\theta_a)a_{100} + sin(\theta_a)e^{i\phi_a}a_{110})}{e^{i\phi_g}(cos(\theta_a)a_{000} + sin(\theta_a)e^{i\phi_a}a_{010})} = \frac{cos(\theta_b)a_{101} + sin(\theta_b)e^{i\phi_b}a_{111}}{cos(\theta_b)a_{001} + sin(\theta_b)e^{i\phi_b}a_{011}}$$

Note that the $e^{i\alpha_a}$, $e^{i\alpha_b}$, $e^{i\phi_g}$ terms cancel out, making $\phi_g$, $\alpha_a$, and $\alpha_b$ free parameters. Multiplying out the denominators, dividng by $cos(\theta_a)cos(\theta_b)$, and re-arranging the terms gives:

$$Atan(\theta_a)tan(\theta_b) + Btan(\theta_a)e^{i\phi_b} + Ctan(\theta_b)e^{i\phi_a} + De^{i(\phi_a+\phi_b)} = 0$$

$$A - Btan(\theta_b)e^{i\phi_b} - Ctan(\theta_a)e^{i\phi_a} + Dtan(\theta_a)tan(\theta_b)e^{i(\phi_a+\phi_b)} = 0$$

Where

$$A = a_{100}a_{001} - a_{000}a_{101}$$
$$B = a_{100}a_{011} - a_{000}a_{111}$$
$$C = a_{110}a_{001} - a_{010}a_{101}$$
$$D = a_{110}a_{011} - a_{010}a_{111}$$

Dividng the first equation by $e^{i(\phi_a+\phi_b)}$ and taking its conjugate results in the following system of equations:

$$A^*tan(\theta_a)tan(\theta_b)e^{i(\phi_a+\phi_b)} + B^*tan(\theta_a)e^{i\phi_a} + C^*tan(\theta_b)e^{i\phi_b} + D^* = 0$$
$$A - Btan(\theta_b)e^{i\phi_b} - Ctan(\theta_a)e^{i\phi_a} + Dtan(\theta_a)tan(\theta_b)e^{i(\phi_a+\phi_b)} = 0$$

Performing the substitution:

$$x = tan(\theta_a)e^{i\phi_a}$$
$$y = tan(\theta_b)e^{i\phi_b}$$

Results in the system of equations:

$$A^*xy + B^*x + C^*y + D^* = 0$$

$$A - By - Cx + Dxy = 0$$

Which can be easily solved by substitution. Once $x$ and $y$ are known, $\theta_a$, $\theta_b$, $\phi_a$, $\phi_b$ can be determined.

Next, the constraint on the eigenvalues of $ab^\dagger$ needs to be satisfied. Using the same parameterization as before, we can write:

$$a = e^{i\phi_g}\begin{bmatrix} cos(\theta_a) & -sin(\theta_a)e^{i\phi_a} \\ sin(\theta_a)e^{i\alpha_a} & cos(\theta_a)e^{i(\phi_a+\alpha_a)} \end{bmatrix} = e^{i\phi_g}\begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha_a} \end{bmatrix}\begin{bmatrix} cos(\theta_a) & -sin(\theta_a)e^{i\phi_a} \\ sin(\theta_a) & cos(\theta_a)e^{i\phi_a} \end{bmatrix}$$

$$b = \begin{bmatrix} cos(\theta_b) & -sin(\theta_b)e^{i\phi_b} \\ sin(\theta_b)e^{i\alpha_b} & cos(\theta_b)e^{i(\phi_b+\alpha_b)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha_b} \end{bmatrix}\begin{bmatrix} cos(\theta_b) & -sin(\theta_b)e^{i\phi_b} \\ sin(\theta_b) & cos(\theta_b)e^{i\phi_b} \end{bmatrix}$$

$$ab^\dagger = e^{i\phi_g}\begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha_a} \end{bmatrix}\left(\begin{bmatrix} cos(\theta_a) & -sin(\theta_a)e^{i\phi_a} \\ sin(\theta_a) & cos(\theta_a)e^{i\phi_a} \end{bmatrix}\begin{bmatrix} cos(\theta_b) & -sin(\theta_b)e^{i\phi_b} \\ sin(\theta_b) & cos(\theta_b)e^{i\phi_b} \end{bmatrix}^\dagger\right)\begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha_b} \end{bmatrix}^\dagger$$

The two matrices between the parentheses are unitary matrices, thus, their product is a unitary matrix and can be expressed as:

$$\begin{bmatrix} cos(\theta_a) & -sin(\theta_a)e^{i\phi_a} \\ sin(\theta_a) & cos(\theta_a)e^{i\phi_a} \end{bmatrix}\begin{bmatrix} cos(\theta_a) & -sin(\theta_a)e^{i\phi_a} \\ sin(\theta_a) & cos(\theta_a)e^{i\phi_a} \end{bmatrix}^\dagger = e^{i\alpha_g}\begin{bmatrix} cos(\theta) & -sin(\theta)e^{i\phi} \\ sin(\theta)e^{i\alpha} & cos(\theta)e^{i(\alpha+\phi)} \end{bmatrix}$$

For some $\alpha_g$, $\alpha$, $\phi$, and $\theta$. Then, we can write

$$ab^\dagger = e^{i\phi_g}\begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha_a} \end{bmatrix}e^{i\alpha_g}\begin{bmatrix} cos(\theta) & -sin(\theta)e^{i\phi} \\ sin(\theta)e^{i\alpha} & cos(\theta)e^{i(\alpha+\phi)} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha_b} \end{bmatrix}^\dagger$$

$$= e^{i(\phi_g+\alpha_g)}\begin{bmatrix} cos(\theta) & -sin(\theta)e^{i(\phi-\alpha_b)} \\ sin(\theta)e^{i(\alpha+\alpha_a)} & cos(\theta)e^{i(\phi+\alpha+\alpha_a-\alpha_b)} \end{bmatrix}$$

The characteristic polynomial of this matrix is:

$$p(t) = (cos(\theta)e^{i(\phi_g+\alpha_g)} - t)(cos(\theta)e^{i(\phi+\alpha+\alpha_a-\alpha_b+\phi_g+\alpha_g)} - t)$$

$$-(-sin(\theta)e^{i(\phi-\alpha_b+\phi_g+\alpha_g)})(sin(\theta)e^{i(\alpha-\alpha_a+\phi_g+\alpha_g)})$$
$$= t^2 - e^{i(\phi_g+\alpha_g)}cos(\theta)(1 + e^{i(\phi+\alpha+\alpha_a-\alpha_b)})t + cos(\theta)^2e^{i(2(\phi_g+\alpha_g)+\phi+\alpha+\alpha_a-\alpha_b)}$$

$$+sin(\theta)^2 e^{i(2(\phi_g+\alpha_g)+\phi+\alpha+\alpha_a-\alpha_b)}$$
$$= t^2 - e^{i(\phi_g+\alpha_g)}cos(\theta)(1+e^{i(\phi+\alpha+\alpha_a-\alpha_b)})t + e^{2i(\phi_g+\alpha_g)}e^{i(\phi+\alpha+\alpha_a-\alpha_b)}$$

We require the eigenvalues to be $\pm 1$, so the characteristic polynomial must be equal to $t^2 - 1$. This requires:

$$e^{i(\phi_g+\alpha_g)}cos(\theta)(1+e^{i(\phi+\alpha+\alpha_a-\alpha_b)}) = 0$$
$$e^{2i(\phi_g+\alpha_g)}e^{i(\phi+\alpha+\alpha_a-\alpha_b)} = -1$$

Solving these equations gives:

$$\phi + \alpha + \alpha_a - \alpha_b = \pi$$
$$\phi_g + \alpha_g = 0$$

Therefore, we set $\phi_g = -\alpha_g$, $\alpha_a = \pi - \phi - \alpha$, and $\alpha_b = 0$ to ensure that $ab^\dagger$ is similar to $X$. Now, all the parameters in $a$ and $b$ are specified, and the procedure from the previous section can be used to compute the gate sequence implementing the desired transformation. This concludes the description of each individual step in our linear nearest neighbors implementation of optimal, exact three-qubit state preparation.

## 6.2 Computing CXdist and CXdist$_1$

We first show how to compute CXdist$_1$ for linear nearest neighbor architecture. For each integer $x$, define:

$$top(x) = max(i),\ s.t.\ 2^i \leq x$$

$$holes(x) = |\{0 \leq k \leq top(x) \text{ and } x >> k \text{ is even}\}|$$

If $x$ is written in its binary representation, then $top(x)$ is the largest position where $x$ has a 1, and $holes(x)$ is the number of zeroes to the right of position $top(x)$ in $x$.

We claim that CXdist$_1(x) = top(x) + holes(x)$. To prove that this is the case, we show that

1. It is always possible to transform $|x\rangle$ to $|1\rangle$ using $top(x) + holes(x)$ CX gates.
2. If a CX gate transforms $|x\rangle$ to $|x'\rangle$, then

$$top(x') + holes(x') \geq top(x) + holes(x) - 1$$

The second point shows that the quantity $top(x)+holes(x)$ decreases by at most 1 for each CX gate, therefore it takes at least $top(x) + holes(x)$ CX gates to transform $|x\rangle$ to a state $|x'\rangle$, where $top(x') + holes(x') = top(1)+holes(1) = 0$. This implies CXdist$_1(x) \geq top(x)+holes(x)$. Taken together with the first statement, they together imply the claim that CXdist$_1(x) = top(x) + holes(x)$.

To prove the first statement, we construct a list $l$ of CX gates that transforms $|x\rangle$ to $|1\rangle$. Let $|c\rangle$ be the state achieved after applying each gate in $l$ to $|x\rangle$. At the start, $|c\rangle = |x\rangle$; at the end, $|c\rangle = |1\rangle$. Construct $l$ as follows:

1. For $i$ in $[top(x), top(x)-1, ..., 1]$, if the $i-1$th bit in $|c\rangle$ is zero, then add CX($i$, $i-1$) to $l$ and update $|c\rangle$. Otherwise do nothing.
2. For $i$ in $[top(x), top(x)-1, ..., 1]$, add CX($i-1$, $i$) to $l$ and update $|c\rangle$.

The first step of the algorithm creates a cascade of 1's starting from the $top(x)$ bit and moving to the right, transforming $c$ into a bitstring of all 1. Then, the second step of the algorithm repeatedly flips the left-most 1 in the binary representation of $c$ until $c = 1$. Thus, this algorithm converts $|x\rangle$ to $|1\rangle$. In addition, the first step uses one CX gate for every zero to the right of $top(x)$, or $holes(x)$ CX gates, while the second step uses $top(x)$ CX gates, for a total of $top(x) + holes(x)$ CX gates. This proves the first statement.

To prove the second statement, we show that if $top(x') = top(x)$, then $|holes(x) - holes(x')| \leq 1$ and if $top(x') \neq top(x)$, then $top(x') \geq top(x) - 1$ and $holes(x) = holes(x')$. Applying a CX gate to $x$ can change at most one bit in its binary representation, so if $top(x)$ is unchanged, then $holes(x)$ can change by at most 1. Next, consider the possible ways a CX gate can modify $top(x)$: either a zero bit to the left of

$top(x)$ is converted to a one, or the one bit at $top(x)$ is converted to a zero. In the former case, given the nearest-neighbor connectivity restriction, the CX gate applied must have used $top(x)$ as its control qubit, targeting qubit $top(x) + 1$. This transformation causes $top(x)$ to increase by 1, but $holes(x)$ is unchanged. Otherwise, the CX gate applied must have used $top(x) - 1$ as the control qubit, and $top(x)$ was the target qubit. This implies the bit at $top(x) - 1$ is a one, therefore, this transformation decreases $top(x)$ by 1 but keeps $holes(x)$ the same. In all cases, either $holes(x)$ decreases by at most 1 while $top(x)$ stays the same, or $top(x)$ decreases by at most 1, leaving $holes(x)$ the same, thus, the quantity $top(x) + holes(x)$ can decrease by at most one for each CX gate applied. This concludes the proof that $\text{CXdist}_1(x) = top(x) + holes(x)$.

To compute CXdist($x$), define:

$$bot(x) = max(c) \ s.t. \ \frac{x}{2^c} \text{ is an integer}$$

$$holes^*(x) = holes(x) - bot(x)$$

If $x$ is written in its binary representation, then $bot(x)$ is the lowest index where $x$ has a 1, and $holes^*(x)$ is the number of zero bits between $top(x)$ and $bot(x)$.

Then we claim:

$$\text{CXdist}(x) = top(x) - bot(x) + holes^*(x)$$

To prove this, we show:

1. It is always possible to transform $|x\rangle$ to $|2^{bot(x)}\rangle$ using $top(x) - bot(x) + holes^*(x)$ CX gates.
2. If a CX gate transforms $|x\rangle$ to $|x'\rangle$, then

$$top(x') - bot(x') + holes^*(x') \geq top(x) - bot(x) + holes^*(x) - 1$$

The second statement shows that the quantity $top(x) - bot(x) + holes^*(x)$ can decrease by at most 1 each time a CX gate is applied. Since this quantity is zero for all powers of 2, it takes at least $top(x) - bot(x) + holes^*(x)$ CX gates to transform $|x\rangle$ to a power of 2. The first statement shows this can always be done, so the statements taken together prove CXdist(x) = $top(x) - bot(x) + holes^*(x)$.

To prove the first statement, we construct a list $l$ of CX gates that transforms $|x\rangle$ to $|2^{bot(x)}\rangle$. Let $|c\rangle$ be the state after applying each gate in $l$ to $|x\rangle$. At the start of the construction, $|x\rangle = |c\rangle$; at the end, $|c\rangle = |2^{bot(x)}\rangle$.

1. For $i$ in $[top(x), top(x) - 1, ..., bot(x) + 1]$, if the $i - 1$th bit in $c$ is zero, then add CX($i$, $i - 1$) to $l$ and update $|c\rangle$.
2. For $i$ in $[top(x), ..., bot(x) + 1]$, add CX($i - 1$, $i$) to $l$ and update $|c\rangle$.

The sequence of CX gates in the first step flips all bits in $c$ between $top(x)$ and $bot(x)$ to 1, and the sequence of CX gates in the second step flips all bits in $c$ between $bot(x) + 1$ and $top(x)$ to 0, thus transforming $|x\rangle$ to $|2^{bot(x)}\rangle$. The first step requires $holes^*(x)$ CX gates while the second step requires $top(x) - bot(x)$ CX gates, for the promised total of $top(x) - bot(x) + holes^*(x)$ CX gates, proving the first statement.

To prove the second statement, we argue that if applying a CX gate to $|x\rangle$ generates the state $|x'\rangle$, then

- If $top(x') \neq top(x)$, then $|top(x') - top(x)| \leq 1$, $bot(x') = bot(x)$, and $holes^*(x') = holes^*(x)$.
- If $bot(x') \neq bot(x)$, then $bot(x') - bot(x)| \leq 1$, $top(x') = top(x)$, and $holes^*(x') = holes^*(x)$.
- If $top(x') = top(x)$ and $bot(x') = bot(x)$, then $|holes^*(x') - holes^*(x)| \leq 1$.

To prove the first point, we recycle the observation from before that $top(x') \neq top(x)$ implies $|top(x') - top(x)| = 1$ and $|holes(x') - holes(x)| = 0$. Thus, all that's left is to show $bot(x') = bot(x)$. This can be seen by noting if $top(x') > top(x)$, then the flipped bit was to the left of $top(x)$, which is to the left of $bot(x)$; otherwise the flipped bit was $top(x)$ and the bit at $top(x) - 1$ is a one, which implies $bot(x) \leq top(x) - 1$, thus $bot(x)$ is to the right of the flipped bit. In both cases, the changed bit is to the left of $bot(x)$, thus, $bot(x) = bot(x')$.

To prove the second point, note that it's a restatement of the first point but with the bits in reverse order. Thus, the symmetric argument of the first point shows the second point.

Finally, if $top(x)$ and $bot(x)$ are the same, we recycle the observation from before that a CX gate flips at most one bit, therefore $holes(x')$ differs from $holes(x)$ by at most one, which implies $holes^*(x')$ differs from $holes^*(x)$ by at most one (because $bot(x') = bot(x)$.

In all cases, at most one of $\{top(x) - top(x'), bot(x) - bot(x'), holes^*(x) - holes^*(x')\}$ will be nonzero, and that value will have absolute value less than one, showing that $top(x) - bot(x) + holes^*(x)$ will decrease by at most one for each CX gate applied. This completes the proof that $\text{CXdist}(x) = top(x) - bot(x) + holes^*(x)$.

## 6.3 CX gate count for UCG exact method on linear nearest neighbor architecture

Bergholm et al [7] showed that exact quantum state preparation on $n$ qubits can be reduced to a sequence of $n$ uniformly controlled single-qubit gates, where the $k$th gate uses qubits $[0...n-1-k]$ as the controls, and qubit $n-k$ as the target. The last UCG is a single-qubit rotation on qubit 0. Next, the authors showed that, on linear nearest neighbor architectures, a uniformly controlled single-qubit gate using controls $[0...c-1]$ and target qubit $c$ can be implemented, up to a diagonal phase gate, as:

$$UCG(c) = \prod_{i=0}^{2^c-1} \text{CX chain}(c-1-z(i), c)U(i)$$

For some choice of single-qubit rotations $U(i)$, where $z(i)$ is the number of trailing zeroes in the binary representation of $i$ (define $z(0) = c-1$, $\frac{i}{2^{z(i)}}$ is an odd integer for $i > 0$) and CX chain$(i,\ j)$ is a sequence of CX gates:

$$\begin{aligned}\text{CX chain}(i,j) = CX(i,i+1)CX(i+1,i+2)...CX(j-2,j-1)CX(j-1,j)\\ CX(j-2,j-1)...CX(i+1,i+2)CX(i,i+1) \quad (2)\end{aligned}$$

The number of CX gates in CX chain$(i,\ j)$ is $2(j-i)-1$. Then, the number of CX gates needed to implement a $UCG(c)$ is

$$\begin{aligned}\text{CX count(UCG}(c)) &= \sum_{i=0}^{2^c-1} (2(1+z(i))-1)\\ &= \sum_{i=0}^{2^c-1} (2z(i)+1)\\ &= 2^c + 2\sum_{i=0}^{2^c-1} z(i)\end{aligned}$$

Since $i$ takes on all nonnegative integers with $c$ bits in the sum, there are $2^{c-m-1}$ instances where $z(i) = m$, for $0 \le m \le c-2$. By contrast, $m = c-1$ will show up twice because $z(0)$ is defined to be $c-1$. Then,

$$\sum_{i=0}^{2^c-1} z(i) = \sum_{m=0}^{c-2} m2^{c-m-1} + 2(c-1) = 2^c - 2$$

$$\text{CX count(UCG}(c)) = 2^c + 2(2^c - 2) = 3 \times 2^c - 4$$

This would be the number of CX gates required to implement UCG$(c)$ on a linear nearest neighbor architecture, however, [7] introduces an optimization: by swapping qubits $c$ and $c-1$ all of the CX chain$(c-1-z(i), c)$ terms in eqn. (2), where $c-1-z(i) < c-1$ will become CX chain$(c-1-z(i), c-1)$. This leads to a cost reduction of 2 CX gates for each of these terms. In addition, for the CX chains where $c-1-z(i) = c-1$, CX chain$(c-1-z(i), c)$ = CX chain$(c-1, c)$ becomes CX chain$(c, c-1)$, which is the same thing. The swapping procedure costs 6 CX gates (3 to swap the qubits, 3 to swap them back at the end), but it saves

2 CX gates for each term where $c-1-z(i) < c-1 \implies z(i) > 0$. There are $2^{c-1}$ such terms, leading to a net savings of $2^c - 6$ CX gates. Thus, we modify the CX count for UCG($c$) gates:

$$\text{CX count, optimized(UCG}(c)) = 3 \times 2^c - 4 - (2^c - 6) = 2 \times 2^c + 2$$

We introduce one additional optimization: the UCG(2), UCG(1), and UCG(0) gates at the end can be replaced by the exact, optimal three-qubit state preparation procedure. Then, we can compute the total number of CX gates required to prepare an $n$-qubit quantum state ($n > 3$) using the UCG method:

$$\begin{aligned}\text{CX count, UCG on } n \text{ qubits} &= \sum_{k=3}^{n-1} 2 \times 2^k + 2 + 3 \\ &= 2(2^n - 8) + 2(n-3) + 3 = 2 \times 2^n + 2n - 19\end{aligned}$$

.

# References


[1] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[2] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[3] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015.

[4] V.V. Shende, S.S. Bullock, and I.L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[5] Kaining Zhang, Min-Hsiu Hsieh, Liu Liu, and Dacheng Tao. Toward trainability of quantum neural networks. *arXiv preprint arXiv:2011.06258*, 2020.

[6] Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2023.

[7] Ville Bergholm, Juha J Vartiainen, Mikko Möttönen, and Martti M Salomaa. Quantum circuits with uniformly controlled one-qubit gates. *Physical Review A*, 71(5):052330, 2005.

[8] Martin Plesch and Časlav Brukner. Quantum-state preparation with universal gate decompositions. *Phys. Rev. A*, 83:032302, Mar 2011.

[9] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. Quantum state preparation with optimal circuit depth: Implementations and applications. *Physical Review Letters*, 129(23):230504, 2022.

[10] Fereshte Mozafari, Giovanni De Micheli, and Yuxiang Yang. Efficient deterministic preparation of quantum states using decision diagrams. *Phys. Rev. A*, 106:022617, Aug 2022.

[11] Michael Ben-Or and Avinatan Hassidim. Fast quantum byzantine agreement. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 481–485, New York, NY, USA, 2005. Association for Computing Machinery.

[12] Daniel Sierra-Sosa, Michael Telahun, and Adel Elmaghraby. Tensorflow quantum: Impacts of quantum state preparation on quantum machine learning performance. *IEEE Access*, 8:215246–215255, 2020.

[13] Niels Gleinig and Torsten Hoefler. An efficient algorithm for sparse quantum state preparation. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, page 433–438. IEEE Press, 2021.

[14] Emanuel Malvetti, Raban Iten, and Roger Colbeck. Quantum Circuits for Sparse Isometries. *Quantum*, 5:412, March 2021.

[15] Sergey Bravyi, Oliver Dial, Jay M Gambetta, Darío Gil, and Zaira Nazario. The future of quantum computing with superconducting qubits. *Journal of Applied Physics*, 132(16), 2022.

[16] Yuichi Hirata, Masaki Nakanishi, Shigeru Yamashita, and Yasuhiko Nakashima. An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture. In *2009 Third International Conference on Quantum, Nano and Micro Technologies*, pages 26–33, 2009.

[17] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, Oct 2010.

[18] Kouhei Nakaji, Shumpei Uno, Yohichi Suzuki, Rudy Raymond, Tamiya Onodera, Tomoki Tanaka, Hiroyuki Tezuka, Naoki Mitsuda, and Naoki Yamamoto. Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators. *Phys. Rev. Res.*, 4:023136, May 2022.

[19] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98:032309, Sep 2018.

[20] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), Mar 2021.

[21] Jarrod R. McClean, Sergio Boixo, Vadim Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), Nov 2018.

[22] Javier Rivera-Dean, Patrick Huembeli, Antonio Acín, and Joseph Bowles. Avoiding local minima in variational quantum algorithms with neural networks. *arXiv preprint arXiv:2104.02955*, 2021.

[23] Thomas J Maldonado, Johannes Flick, Stefan Krastanov, and Alexey Galda. Error rate reduction of single-qubit gates via noise-aware decomposition into native gates. *Scientific Reports*, 12(1):6379, 2022.

[24] Marko Žnidarič, Olivier Giraud, and Bertrand Georgeot. Optimal number of controlled-not gates to generate a three-qubit state. *Phys. Rev. A*, 77:032320, Mar 2008.

[25] Harper R Grimsley, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature communications*, 10(1):3007, 2019.

[26] Israel F. Araujo, Ismael C. S. Araújo, Leon D. da Silva, Carsten Blank, and Adenilton J. da Silva. Quantum computing library, February 2023.