An Efficient Algorithm for Sparse Quantum State Preparation

Niels Gleinig
Department of Computer Science
ETH Zurich
Zurich, Switzerland
niels.gleinig@inf.ethz.ch

Torsten Hoefler
Department of Computer Science
ETH Zurich
Zurich, Switzerland
torsten.hoefler@inf.ethz.ch

***Abstract*—Generating quantum circuits that prepare specific states is an essential part of quantum compilation. Algorithms that solve this problem for general states generate circuits that grow exponentially in the number of qubits. However, in contrast to general states, many practically relevant states are sparse in the standard basis. In this paper we show how sparsity can be used for efficient state preparation. We present a polynomial-time algorithm that generates polynomial-size quantum circuits (linear in the number of nonzero coefficients times number of qubits) that prepare given states, making computer-aided design of sparse state preparation scalable.**

***Index Terms*—Quantum Computing, Quantum Compilation, State Preparation, Circuit Synthesis**

I. Introduction

The discovery of quantum algorithms that solve certain problems asymptotically faster than any known classical algorithms [1] [2] indicate that quantum computers may be much more powerful than classical computers. Quantum compilation, as an interface between the algorithmic high-level and technological low-level aspects of quantum computing, encapsulates various logic synthesis tasks. Some of these synthesis tasks are similar to the synthesis tasks in the design of classical processors and hence allow similar optimization techniques. Many of those have been investigated by the computer-aided design (CAD) community [3] [4] [5]. Yet, due to the special phenomena of superposition and entanglement, certain parts of the quantum compilation process are very different from the analogous tasks in classical computing and do not allow us to transfer knowledge.

One of those tasks is quantum state preparation, the process of creating a specific initial quantum state that may encode input data or a problem instance. State preparation is an essential part of quantum computing and much more complex than the analogue in classical computing [6]. While in classical computing all possible states of a memory-block of size $O(n)$ can be obtained by setting the appropriate bits using $O(n)$ NOT-gates, the possible states of a quantum computer are superpositions of basis states and some of them can only be created with very complex circuits [7] [8]. In fact, it has been shown that using a gate library of only one-qubit-gates and CNOT-gates, some of the states of an n-qubit quantum computer can only be prepared with circuits of size $\Omega(2^n)$ [7] [8]. Hence, state preparation methods that work for general states are unlikely to scale beyond very few qubits.

However, in contrast to general states, many (if not most) practically relevant states have the property that only a small proportion of the basis states have nonzero coefficients (we will explain the concepts of basis states later in more detail). We call states with this property *sparse*. To name a few prominent examples, the proportion of basis states with nonzero coefficient among all basis states is $O(1/\sqrt{2^n})$ for generalized Bell states and thermofield double states [9], $O(n/2^n)$ for W-states [10], and $O(1/2^n)$ for GHZ-states. Further, in many quantum algorithms we need to prepare states that encode a problem instance and depending on the problem instance these states are sparse too. Examples of this include *quantum linear system solvers* [2].

In this paper we show how sparsity can be leveraged to make state preparation asymptotically more efficient than preparation of general states. Our main contribution is an algorithm that takes a quantum state ϕ with $|S|$ nonzero coefficients as input, and produces in $O(|S|^2 \log(|S|)n)$ classical runtime a circuit C of $O(|S|n)$ size (having $O(|S|n)$ CNOT gates and $O(|S| \log(|S|) + n)$ single qubit gates), such that C maps the initial state $|0^n\rangle$ to ϕ.

A. Background and Problem Formulation

1) (Sparse) quantum states: The *basis states* of an n-qubit quantum system are a set of 2^n orthogonal vectors that can be naturally identified with boolean n-bit strings: $|0\dots00\rangle$,$|0\dots01\rangle$,...,$|1\dots11\rangle$. A general state ϕ of an n-qubit quantum system is an arbitrary normalized linear combination of basis states. That is,

$$\phi = \sum_{x\in\{0,1\}^n} c_x |x\rangle , \sum_{x\in\{0,1\}^n} |c_x|^2 = 1. \tag{1}$$

Hence, in general, describing an n-qubit quantum state classically requires specifying all 2^n *amplitudes* c_x. However, in many interesting quantum states, most of the coefficients c_x are zero. We call quantum states with this property *sparse quantum states*. Letting $S \subset \{0,1\}^n$ denote the set of basis states with nonzero coefficients, we have

$$\phi = \sum_{x\in S} c_x |x\rangle . \tag{2}$$

Hence, if $|S| << 2^n$ it is more efficient to specify ϕ by only storing the set S and the coefficients $c_x, x \in S$.

2) Quantum gates and quantum circuits: In order to transform one state into another, we need to apply a sequence of elementary gates from a given gate library to it. In this paper we consider a gate library that consists of the following types of gates:

- One-qubit-gates: For a given single-qubit-state $\alpha|0\rangle + \beta|1\rangle$ we let $G_{\alpha|0\rangle+\beta|1\rangle}$ denote any one-qubit-transformation that maps $\alpha|0\rangle + \beta|1\rangle$ to $e^{i\lambda}|0\rangle$.
- CNOT-gates: these have one control qubit and one target qubit. Applied on a basis state $|x\rangle$, they act by flipping the target bit of the boolean string x iff the control bit is set. Their action extends to general superpositions of basis states through linearity.

A quantum circuit is a sequence of quantum gates. For the sake of readability, we will use in our circuit constructions multicontrolled operations. However, when we make statements about the size of our circuits, we refer to the number of gates used in the circuit when decomposing it (including the multicontrolled operations) into a sequence of the elementary gates described above.

3) Problem formulation: **Sparse Quantum State Preparation:** Given the set $S \subset \{0,1\}^n$ and the nonzero coefficients $c_x, x \in S$ that describe a quantum state ϕ, find a sequence of elementary quantum gates $g_1, g_2, \ldots, g_k$, such that applying the quantum circuit $C := g_1 g_2 \ldots g_k$ to $|0^n\rangle$ creates $C|0^n\rangle = \phi$.

Obviously, we would like the circuit C to be as small as possible. Notice that this problem is equivalent to finding a circuit $C' := g'_1 g'_2 \ldots g'_k$, such that $C'\phi = |0^n\rangle$, because setting $C = (g'_k)^{-1}(g'_{k-1})^{-1}\ldots(g'_1)^{-1}$ we have $C|0^n\rangle = \phi$. Given this direct equivalence of the problems, in the remainder of this paper we will use the problem formulation that asks for finding a circuit C that transforms ϕ to $|0^n\rangle$.

B. Previous work

The problems of synthesizing and optimizing quantum circuits that compute classical functions has been widely investigated by the CAD community [11] [12] [13] [3] [4] [5] [14]. For a long time, the problem of synthesizing quantum circuits that prepare certain states has been investigated from a rather theoretical point of view [15] [6]. In this theoretical context, the main interest is to determine which states can be prepared by circuits of polynomial size. It was known that superpositions of polynomialy many basis states can be prepared by circuits of polynomial size [6]. However, the theoretical investigations often disregard the exact degrees of the polynomials and mainly ask for existence of polynomial-size circuits rather than providing concrete, optimized constructions. Hence, the theoretical insights have had no influence on the CAD methods developed so far: all existing CAD methods are designed for general states without adapting to the complexity class of the given state. Shende et al. [16] discuss a method for preparing general states using $O(2^n)$ gates. Möttönen et al. [8] discuss another method for preparing general quantum states using $O(2^n)$ CNOT gates and $O(2^n)$ single-qubit-gates. The work of Kaye and Mosca [17] presents a method for preparing quantum states with non-negative, real coefficients using circuits of exponential size. Also the method proposed by Niemann et al. [18] uses $O(2^n)$ CNOT gates and $O(2^n)$ single-qubit-gates. Mozafari et al. [19] present a method for preparing uniform quantum states, that is, states in which all nonzero coefficients are equal. Their method uses $O(2^n)$ CNOT gates and $O(2^n)$ single-qubit-gates. Araujo et al. [20] propose another method for state preparation. Interestingly, their method uses circuits of only linear depth. But this comes at the prize of using an exponential amount of garbage qubits, and still, the total amount of used gates is $\Omega(2^n)$.

Overall, there exist many interesting approaches to synthesizing circuits that prepare general quantum states. However, none of the CAD methods that solve this problem for general states produces circuits of size less than $O(2^n)$. Although some of these methods propose instance specific optimizations that result in small circuits for some specific states, none of them leverages sparsity to produce generally small circuits for sparse states. In fact, these methods will produce even for most sparse states circuits of exponential size because they are based on "uniformly controlled rotations" (uniformly controlled rotations require in general a number of gates exponential in the number of control bits).

Using simple dimensionality arguments (as presented for example in the introduction of Möttönen et al. [8]) one can show that there cannot exist methods that solve this problem for general states using circuits of size less than $O(2^n)$. They show lower bounds of $2^{n+1}-2$ for the number of one-qubit-gates, and $\lceil \frac{1}{4}(2^{n+1} - 3n - 8)\rceil$ for the number of CNOT-gates. This implies the bound

$$Tb(n) := \lceil \frac{1}{4}(5 \cdot 2^{n+1} - 3n - 10)\rceil \tag{3}$$

for the total number of gates. In this sense, the existing methods are even optimal, but they are still too expensive because the problem they solve is too general. Hence, to find efficient state preparation methods we need to restrict the states to some special class of states.

II. The Algorithm

A. Intuitive explanation through examples

If $\phi = |x\rangle$ is equal to a basis state, that is, $x \in \{0,1\}^n$, the problem of transforming ϕ to $|0^n\rangle$ is trivial: We only need to apply NOT-gates to all qubits $i \in \{1,2,\ldots,n\}$ with $x[i] = 1$.

If $\phi = \alpha|x\rangle + \beta|y\rangle$ is the superposition of only two basis states, we first find a qubit $i \in \{1,2,\ldots,n\}$ for which $x[i] \neq y[i]$. Then, for all other qubits $j \in \{1,2,\ldots,n\}$ with $x[j] \neq y[j]$, we apply CNOT-gates that target the qubit j controlled on qubit i. This transforms ϕ into a state $\alpha|\tilde{x}\rangle + \beta|\tilde{y}\rangle$, where $\tilde{x}, \tilde{y} \in \{0,1\}^n$ are strings that only differ in the i-th entry. The i-th qubit is now in a pure state, either $\alpha|0\rangle + \beta|1\rangle$ or $\alpha|1\rangle + \beta|0\rangle$. Applying now to the i-th qubit a gate that transforms this state into $e^{i\lambda}|0\rangle$ (that is, either $G_{\alpha|0\rangle+\beta|1\rangle}$ or $G_{\alpha|1\rangle+\beta|0\rangle}$), the state of the whole system $\alpha|\tilde{x}\rangle + \beta|\tilde{y}\rangle$ becomes either $|\tilde{x}\rangle$ or $|\tilde{y}\rangle$ (the one that has a zero in the i-th entry). So we have transformed ϕ into a basis state and can now proceed as described above.

When ϕ is the superposition of three or more basis states, say $\phi = \alpha|x\rangle + \beta|y\rangle + \gamma|z\rangle$, we encounter a phenomenon that makes the problem more difficult. First, we can still find a sequence of CNOT-gates that transforms this state into a state $\alpha|\tilde{x}\rangle + \beta|\tilde{y}\rangle + \gamma|\tilde{z}\rangle$ such that $\tilde{x}$ and $\tilde{y}$ only differ in one entry i. But now, if we apply to the i-th qubit a gate that "merges" $\alpha|\tilde{x}\rangle + \beta|\tilde{y}\rangle$ into a single basis state as we did above, this same gate may have the effect of "splitting" $\gamma|\tilde{z}\rangle$ into the superposition of two basis states. To circumvent this problem we control this operation on a set of qubits so that it is performed on $|\tilde{x}\rangle$ and $|\tilde{y}\rangle$ but not on $|\tilde{z}\rangle$. In general, if $\phi = \sum_{x\in S} c_x|x\rangle$ is the superposition of $|S|$ basis states, to avoid "splitting" basis states we need to control the merging operation in such a way that for only two elements $x, y \in S$ the operation is performed, while all other elements remain invariant. However, the cost of controlled operations grows linearly with the number of control qubits that are used [21]. The key to the efficiency of our method relies on two tricks:

1) We find a way of "merging" $\tilde{x}$ and $\tilde{y}$ with not more than $O(\log(|S|))$ control bits. Setting this up is the main technical difficulty of our algorithm.
2) We choose the gate $G_{\alpha|0\rangle+\beta|1\rangle}$ that maps $\alpha|0\rangle + \beta|1\rangle$ to $e^{i\lambda}|0\rangle$ of the form

$$M = \begin{bmatrix} \sin(\omega) & \mathrm{e}^{i\alpha}cos(\omega) \\ \mathrm{e}^{-i\alpha}cos(\omega) & \text{-}\sin(\omega) \end{bmatrix}.$$

Gates of this form can be multi-controlled efficiently: we can implement this as a sequence of two one-qubit-gates and a multi-controlled Tofoli gate [21], and implement the multi-controlled Tofoli gate with a number of CNOT-gates linear in the number of control bits [22], adding up to $O(\log(|S|)$ elementary gates.

B. Formal description

First, we describe a method to produce a circuit that transforms a superposition of $|S|$ basis states into a superposition of less than $|S|$ basis states. The pseudocode of this method is shown

in Algorithm 1. The following lemmas proof the correctness and efficiency of Algorithm 1.

Lemma 1. *Given a state $\phi = \sum_{x\in S} c_x |x\rangle$, Algorithm 1 produces a circuit C such that $C\phi = \sum_{x\in S'} c'_x |x\rangle$ with $|S'| < |S|$.*

Proof. Notice that after each iteration through the first WHILE-loop the elements in T are exactly those strings $x \in S$ that have on the bits dif_qubits the entry-values from dif_values (that is, $x[dif\_qubits[i]] = dif\_values[i]$ for all $1 \leq i \leq |dif\_qubits|$). Hence, the single element contained in T after we finish the WHILE-loop, x_1, is the only string in S that has on the bits dif_qubits the entry-values from dif_values.

Now let l denote the number of iterations we spent at the first WHILE-loop and notice that l is the length of dif_qubits and dif_values. So when we pop the last value of dif_qubits and dif_values (line 14 and 15) and define T' (line 17) as the set of strings that coincide now on the bits dif_qubits with the values from dif_values, this set coincides with the set T that we had at the $(l-1)$-th iteration through the WHILE-loop. As we had $|T| > 1$ at the $(l-1)$-th iteration, T' contains x_1 and at least one more string. So when we remove x_1 from T' (line 18), T' is not empty, and the strings that remain are exactly those that were removed from T after the $(l-1)$-th iteration. Hence, recalling that dif denotes the bit that was popped from dif_qubits (and hence the one added at the last iteration), we have $x_1[dif] \neq x'[dif]$ for all $x' \in T'$.

After finishing the second WHILE-loop the only string $x_2 \in T'$ is the only string in S that coincides on the bits dif_qubits with the values from dif_values, except maybe x_1. Yet we have $x_1[dif] \neq x_2[dif]$. So when we apply the dif-controled CNOT-gates in lines 32 to 36 we make x_1 and x_2 become equal on all bits except dif.

In particular, now x_1 and x_2 equal on all the bits from dif_qubits and, as we will now show, no other strings in S equal with x_1 and x_2 on these bits. To proof this latter assertion, we consider any $y \in S$ with $y \neq x_1, y \neq x_2$ and make a case distinction: The string y was either removed from T in one of the first $l-1$ iterations of the first WHILE-loop, or it was removed in the last iteration of the first WHILE-loop. In the first case, y differed (before the CNOT-gates were added) from x_1 and x_2 on one of the first $l-1$ entries in dif_qubits. But since x_1 and x_2 equal on all of the first $l-1$ entries from dif_qubits we did not apply any CNOT-gate targeting any of those bits and hence y still differs from x_1 and x_2 on this bit after the CNOT-gates were applied. In the latter case we have $y[dif] \neq x_1[dif]$ and hence for the CNOT-gates that are added the control bit is not set in y. But since y differs on at least one of the bits in dif_qubits from x_2, it will still differ on this bit after the CNOT-gates are applied. By applying the NOT-gates in line 39 we ensure that all bits in dif_qubits are set for x_1 and x_2 but they are not all set for any other $y \in S$. Due to the NOT-gate that we (possibly) added in line 30 we ensure that $x_1[dif] = 1$ and $x_2[dif] = 0$, and hence the gate $G_{c_{x_1}|1\rangle+c_{x_2}|0\rangle}$ in line 42 "merges" x_1 and x_2. And since the operation in line 42 is controlled on the bits in dif_qubits it will only be applied to x_1 and x_2 but no other $y \in S$. Hence, it does not "split" any other $y \in S$, as we wanted. □

Lemma 2. *Let C denote the circuit that Algorithm 1 outputs. Using a primitive gate library of one-qubit-gates and CNOT-gates, we can implement C with $O(n)$ CNOT-gates and $O(\log(|S|))$ one-qubit-gates.*

Proof. To bound the size of $|dif\_qubits|$, observe that in each iteration of the WHILE-loops the sizes of $|T|$ and $|T'|$ are halved. Hence $|dif\_qubits| \leq \lceil \log_2(|S|) \rceil + 1$.

ALGORITHM 1: Input: The classical specification of an n-qubit quantum state $\phi = \sum_{x\in S} c_x |x\rangle$ (given by the set $S \subset \{0,1\}^n$ and the list of coefficients $c_x, x \in S$). Output: A circuit C such that $C\phi = \sum_{x\in S'} c'_x |x\rangle$ with $|S'| < |S|$.

1: Initialize an empty quantum circuit C;
2: Initialize an empty stack $dif\_qubits = []$; // This is a stack of bits $b \in \{1, 2, \ldots, n\}$ that will hold in the end the bits that we use as control for the "merging" step
3: Initialize an empty stack $dif\_values = []$; // This is a stack of boolean values
4: Initialize the set $T = S$;
5: **while** $|T| > 1$ **do**
6: Find a qubit $b \in \{1, 2, \ldots, n\}$ such that the sizes of the sets $T_0 := \{x \in T | x[b] == 0\}$ and $T_1 := \{x \in T | x[b] == 1\}$ are as unequal as possible but neither set is empty;
7: Append b to dif_qubits;
8: **if** $|T_0| < |T_1|$ **then**
9: Set $T = T_0$ and append 0 to dif_values;
10: **else**
11: Set $T = T_1$ and append 1 to dif_values;
12: **end if**
13: **end while**
14: Pop the last value appended to dif_qubits and store it as dif;
15: Pop the last value that was appended to dif_values;
16: Store the single element in T as x_1;
17: Let $T' \subset S$ denote the set of strings that have the values in dif_values on the bits dif_qubits;
18: Remove x_1 from T';
19: **while** $|T'| > 1$ **do**
20: Find a qubit $b \in \{1, 2, \ldots, n\}$ such that the sizes of the sets $T'_0 := \{x \in T' | x[b] == 0\}$ and $T'_1 := \{x \in T' | x[b] == 1\}$ are as unequal as possible but neither set is empty;
21: Append b to dif_qubits;
22: **if** $|T'_0| < |T'_1|$ **then**
23: Set $T' = T'_0$ and append 0 to dif_values;
24: **else**
25: Set $T' = T'_1$ and append 1 to dif_values;
26: **end if**
27: **end while**
28: Let x_2 denote the single element in T';
29: **if** $x_1[dif] \neq 1$ **then**
30: Add to C a NOT-gate on line dif;
31: **end if**
32: **for** b in $\{1, 2, \ldots n\} \setminus \{dif\}$ **do**
33: **if** $x_1[b] \neq x_2[b]$ **then**
34: Add to C a CNOT-gate targeting b controled on dif;
35: **end if**
36: **end for**
37: **for** b in dif_qubits **do**
38: **if** $x_2[b] \neq 1$ **then**
39: Add to C a NOT-gate on line b;
40: **end if**
41: **end for**
42: Apply on qubit dif a $G_{c_{x_1}|1\rangle+c_{x_2}|0\rangle}$-gate controlled on the qubits in dif_qubits;
43: **return** C;

In the first FOR-loop we add $O(n)$ CNOT-gates.

In the second FOR-loop we add up to $|dif\_qubits|$ one-qubit-gates, which is $O(\log(|S|))$. Finally, we need to implement $G(c_{x_1}|1\rangle + c_{x_2}|0\rangle)$ with $O(\log(|S|))$ control bits. We can do this with the construction proposed by Barenco et al. [21], requiring two one-qubit-gates and a $(\lceil \log_2(|S|)\rceil + 1)$-control Toffoli gate. Using the method proposed by Gidney [22], we can implement the multi-control Toffoli gates with $O(\log(|S|)$ CNOT-gates. Since $|S| \leq 2^n$, the total number of CNOT-gates is $O(n)$. □

Lemma 3. *The classical runtime of Algorithm 1 is $O(|S|\log(|S|)n)$.*

Proof. The WHILE-loops are the bottleneck. Each iteration of the WHILE-loops takes $O(|S|n)$ time (for each of the n bits we need to look at the $|S|$ strings). As we have seen earlier, the number of iterations is $O(\log(|S|))$. □

Algorithm 1 allows us to build circuits that transform a given superposition of $|S|$ basis states into a superposition of $|S|-1$ basis states. Using it, we are now in the position to present Algorithm 2, the main contribution of this paper.

ALGORITHM 2: Input: The classical specification of an n-qubit quantum state $\phi = \sum_{x\in S} c_x|x\rangle$ (given by the set $S \subset \{0,1\}^n$ and the list of coefficients $c_x, x \in S$). Output: A circuit C such that $C|0^n\rangle = \phi$.

```
Initialize an empty quantum circuit C;
while |S| > 1 do
    Use Algorithm 1 to find a circuit Ĉ such that Ĉφ = Σ_{x∈S'} c'_x
    with |S'| < |S|;
    Update the state φ = Ĉφ;
    Update the circuit C = Ĉ ∘ C;
end while
Add to C the NOT-gates needed to transform φ into |0^n⟩;
Invert the gates from C and reverse their order;
return C;
```

Theorem 1. *The circuit C produced by Algorithm 2 can be implemented with $O(|S|n)$ CNOT-gates and $O(|S|\log(|S|)+n)$ one-qubit-gates. The classical runtime of Algorithm 2 is $O(n|S|^2\log(|S|))$.*

Proof. Observe that running Algorithm 2 is the same as running Algorithm 1 and then running Algorithm 2 for an instance of smaller size. So if we let $T_{CNOT}(|S|)$ denote the maximal number of CNOT-gates used for instances of ϕ with sets S of a given size $|S|$, this function satisfies a recursive relation. Applying Lemma 2 we get

$$T_{CNOT}(|S|) \leq T_{CNOT}(|S|-1) + cn \tag{4}$$

for some $c \in \mathbb{R}$ that is independent of $|S|$ and n. Hence,

$$T_{CNOT}(|S|) \leq \sum_{i=1}^{|S|} cn \in O(|S|n). \tag{5}$$

Defining $T_{\text{1-qubit-gates}}(|S|)$ similarly, we get

$$T_{\text{1-qubit-gates}}(|S|) \leq T_{\text{1-qubit-gates}}(|S|-1) + c\log(|S|). \tag{6}$$

As $T_{\text{1-qubit-gates}}(1) = n$, we get $T_{\text{1-qubit-gates}}(|S|) \in O(|S|\log(|S|)+n)$. Likewise, letting $T(|S|)$ denote the runtime and applying Lemma 3 we get

$$\begin{aligned} T(|S|) &\leq T(|S|-1) + c|S|\log(|S|)n \\ &\leq cn\sum_{i=1}^{|S|} i\log(i) \in O(n|S|^2\log(|S|)). \end{aligned} \tag{7}$$

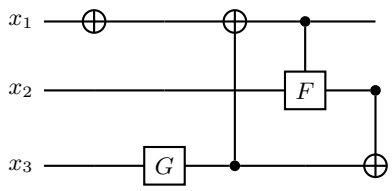

Fig. 1: Quantum circuit to prepare the state $|b\rangle = \frac{1}{\sqrt{168}}(2|001\rangle + 8|100\rangle + 10|111\rangle$, where F is a gate that maps $|0\rangle$ to $\frac{8}{\sqrt{164}}|0\rangle + \frac{10}{\sqrt{164}}|1\rangle$ and G is a gate that maps $|0\rangle$ to $\frac{2}{\sqrt{168}}|0\rangle + \frac{\sqrt{164}}{\sqrt{168}}|1\rangle$.

□

III. Application examples and experimental results

A. Quantum linear system solvers

Harrow et al. [2] presented a quantum algorithm to solve linear systems of the form $A\cdot x = b \in \mathbb{R}^{2^n}$. To encode $b = (b_0, b_1, \ldots, b_{2^n})$, one needs to prepare the state

$$|b\rangle := C\sum b_i|i\rangle, \tag{8}$$

where C is a normalizing constant. Assuming the state $|b\rangle$ is given and the matrix A satisfies certain conditions (conditions that essentially allow e^{iAt} to be applied in time less than the actual size of A; for more details see the reference), this algorithms runs on sparse systems in $O(nk^2)$ time, where k is the condition number. Notice the exponential speedup: Since b has size 2^n it would take $\Omega(2^n)$ time to solve this problem classically, even if b is sparse.

However, this quantum algorithm assumes that an efficient way for preparing $|b\rangle$ is provided. This is problematic. Preparing a general state $|b\rangle$ can require $\Omega(2^n)$ time and becomes the bottleneck. Hence, efficient preparation cannot be assumed for general b when n is large. Using our method we can generate $|b\rangle$ efficiently for vectors b that are sparse.

Example: When $b = (0,2,0,0,8,0,0,10)$, we have

$$|b\rangle = \frac{1}{\sqrt{168}}(2|001\rangle + 8|100\rangle + 10|111\rangle). \tag{9}$$

Applying our algorithm we can generate this state with the circuit depicted in Figure 1. The state corresponding to the sparse vector $b \in \mathbb{R}^{1048576}$ with 8 nonzero entries in the positions $1, 5, 50, 8000, 80001, 1000000, 1000100, 1000200$ is a sparse 20-qubit state with 8 nonzero coefficients. The circuit that our algorithm found to prepare this state had 70 gates.

B. Random states

We test our algorithm on randomly generated sparse states. Throughout this section we let the parameter k denote the number of basis states with nonzero coefficients. To sample random sparse n-qubit states with k nonzero coefficients, we draw uniformly at random k different basis states $|x\rangle, x \in \{0,1\}^n$ and let ϕ be the uniform superposition of those states. Although we let this superposition be uniform (i.e., set the coefficients of the k sampled basis states to be equal), the size of the circuits produced by our algorithm does not depend on the exact value of the coefficients (as long as they are nonzero). Hence, the diagrams in this section would all look equal if we had chosen the coefficients according to some random distribution (e.g., sampling them from a complex normal distribution and then normalizing). To compare the size of the circuits produced by our method to those produced by methods that do not distinguish

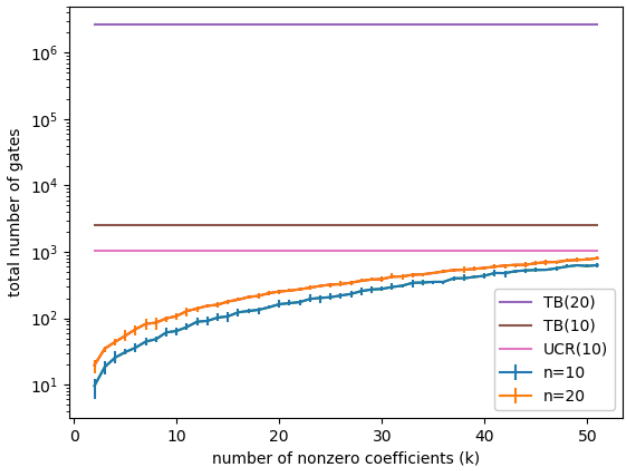

Fig. 2: Increasing nonzero coefficients

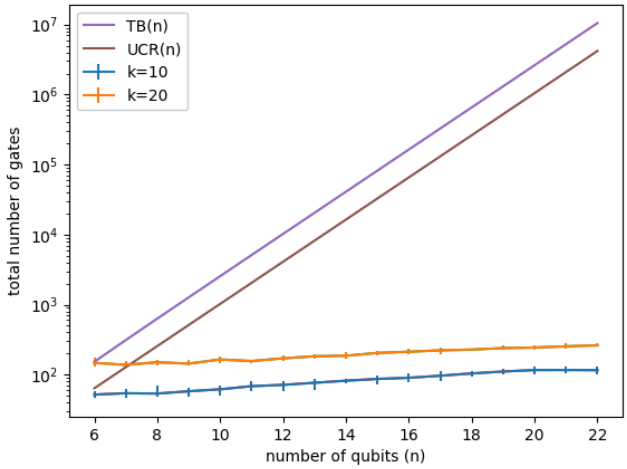

Fig. 3: Increasing qubits

between sparse and non-sparse states we include in the following diagrams $Tb(n)$ as defined by Equation (3), which is a lower bound for the total number of gates needed for general states [8]. We also show the cost of a single "uniformly controlled rotation" on n qubits $UCR(n) := 2^n$, which is the basic building block of most existing state preparation methods [16], [8], [17], [19], [18] (to the best of our knowledge, this number of gates is required even if, as in the case of sparse states, many of their angles are zero). For each combination of parameters shown in the following diagrams, we sampled 10 random states and show the average size of the circuits produced by our algorithm as well as a bar indicating the smallest and largest circuits.

In Figure 2 we see how the size of our circuits grows as we increase k while fixing n at $n = 10$ and $n = 20$. In Figure 3 we see how the size of our circuits grows as we increase the number of qubits n while fixing k at $k = 10$ and $k = 20$. In Figure 4 we see how the size of our circuits grows as we increase the number of qubits n and simultaneously increase k as a function of n. We show the curves for $k = n$ and $k = n^{1.5}$. According to Theorem 1 the growth of these

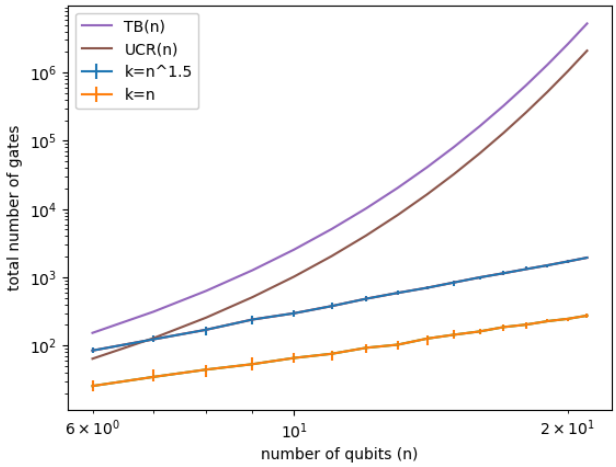

Fig. 4: Increasing n and k as a function of n

State	CNOTS	one-qubits-gates	total
$W(100)$	295	198	493
$W_{3-banded}(100)$	289	196	485
$INC(100)$	196	198	394

Fig. 5: Sizes of circuits produced for special states.

curves should be in $O(n^2)$ and $O(n^{2.5})$ respectively. To verify this polynomial growth we used in this figure logarithmic scales on both axes. The linear shape of the curves confirms that the circuits grow polynomially.

C. Special states

We run our algorithm on the following 3 special 100-qubit states:

$$W(100) := \frac{1}{\sqrt{100}} \sum_{i=1}^{100} |0^{i-1}10^{100-i}\rangle$$
$$W_{3-banded}(100) := \frac{1}{\sqrt{98}} \sum_{i=1}^{98} |0^{i-1}1110^{98-i}\rangle \qquad (10)$$
$$INC(100) := \frac{1}{\sqrt{100}} \sum_{i=1}^{100} |1^i0^{100-i}\rangle\,.$$

Notice that 100 qubits is unfeasible for general state preparation methods: both because of the classical runtime to process 2^{100} complex amplitudes as well as the sizes of the circuits. The sizes of the circuits produced by our algorithm are summarized in Figure 5.

D. Implementation details

We implemented our algorithm in Python and ran the experiments on a 2.5 GHz Intel Core i5. For each state sampled in our experiments, the computation of the circuit was done in less than a second. Using Qiskit, we verified that the circuits indeed produce the correct states.

IV. Conclusions and Future Work

We present a scalable algorithm for computer-aided design of quantum state preparation. The classical runtime of our algorithm is $O(|S|^2 \log(|S|)n)$. For the preparation of states that are the superposition of $|S|$ basis states, our algorithm produces circuits of size $O(|S|n)$. Hence, for states that satisfy the sparsity condition

$|S| \in o(\frac{2^n}{n})$, preparation with our method is asymptotically more efficient than using methods that prepare general states.

ACKNOWLEDGMENT

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 programme (grant agreement DAPP, No. 678880), and by Microsoft Research through its Swiss Joint Research Centre.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, Oct 2009. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.103.150502

[3] D. Maslov and G. W. Dueck, "Reversible cascades with minimal garbage," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 11, pp. 1497–1509, 2004.

[4] M. Soeken, R. Wille, O. Keszocze, D. M. Miller, and R. Drechsler, "Embedding of large boolean functions for reversible logic," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 4, pp. 41:1–41:26, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2786982

[5] A. Zulehner and R. Wille, "Make it reversible: Efficient embedding of non-reversible functions," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, 2017, pp. 458–463. [Online]. Available: https://doi.org/10.23919/DATE.2017.7927033

[6] S. Aaronson, "Multilinear formulas and skepticism of quantum computing," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 118–127. [Online]. Available: https://doi.org/10.1145/1007352.1007378

[7] E. Knill, "Approximation by quantum circuits," 5 1995.

[8] M. Möttönen, J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Transformation of quantum states using uniformly controlled rotations," *Quantum Inf. Comput.*, vol. 5, pp. 467–473, 2005.

[9] W. Cottrell, B. Freivogel, D. M. Hofman, and S. F. Lokhande, "How to build the thermofield double state," *Journal of High Energy Physics*, vol. 2019, no. 2, Feb 2019. [Online]. Available: http://dx.doi.org/10.1007/JHEP02(2019)058

[10] W. Dür, G. Vidal, and J. I. Cirac, "Three qubits can be entangled in two inequivalent ways," *Phys. Rev. A*, vol. 62, p. 062314, Nov 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.62.062314

[11] R. Wille, M. Soeken, and R. Drechsler, "Reducing the number of lines in reversible circuits," pp. 647–652, 2010. [Online]. Available: http://doi.acm.org/10.1145/1837274.1837439

[12] R. Wille, O. Keszocze, and R. Drechsler, "Determining the minimal number of lines for large reversible circuits," *2011 Design, Automation and Test in Europe*, pp. 1–4, 2011.

[13] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler, "Synthesis of reversible circuits with minimal lines for large functions," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. IEEE, 2012, pp. 85–92.

[14] N. Gleinig, F. A. Hubis, and T. Hoefler, "Embedding functions into reversible circuits: A probabilistic approach to the number of lines," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, June 2019, pp. 1–6.

[15] D. Aharonov and A. Ta-Shma, "Adiabatic quantum state generation and statistical zero knowledge," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 20–29. [Online]. Available: https://doi.org/10.1145/780542.780546

[16] V. Shende, S. Bullock, and I. Markov, "Synthesis of quantum-logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, p. 1000–1010, Jun 2006. [Online]. Available: http://dx.doi.org/10.1109/TCAD.2005.855930

[17] P. Kaye and M. Mosca, "Quantum networks for generating arbitrary quantum states," in *Optical Fiber Communication Conference and International Conference on Quantum Information*. Optical Society of America, 2001, p. PB28. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=ICQI-2001-PB28

[18] P. Niemann, R. Datta, and R. Wille, "Logic synthesis for quantum state generation," in *2016 IEEE 46th International Symposium on Multiple-Valued Logic (ISMVL)*, 2016, pp. 247–252.

[19] F. Mozafari, M. Soeken, and G. De Micheli, "Preparation of uniform quantum states utilizing boolean functions."

[20] I. F. Araujo, D. K. Park, F. Petruccione, and A. J. da Silva, "A divide-and-conquer algorithm for quantum state preparation," 2020.

[21] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, p. 3457–3467, Nov 1995. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.52.3457

[22] C. Gidney, "Constructing large controlled nots," 2015 (accessed November 19, 2020), https://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html.