Best Approximate Quantum Compiling Problems

LIAM MADDEN, University of Colorado Boulder
ANDREA SIMONETTO, UMA, ENSTA Paris, Institut Polytechnique de Paris

We study the problem of finding the best approximate circuit that is the closest (in some pertinent metric) to a target circuit, and which satisfies a number of hardware constraints, like gate alphabet and connectivity. We look at the problem in the CNOT+rotation gate set from a mathematical programming standpoint, offering contributions both in terms of understanding the mathematics of the problem and its efficient solution. Among the results that we present, we are able to derive a 14-CNOT 4-qubit Toffoli decomposition from scratch, and show that the Quantum Shannon Decomposition can be compressed by a factor of two without practical loss of fidelity.

CCS Concepts: • **Theory of computation → Mathematical optimization**; *Quantum computation theory*; • **Computer systems organization → Quantum computing;**

ACM Reference format:
Liam Madden and Andrea Simonetto. 2022. Best Approximate Quantum Compiling Problems. *ACM Trans. Quantum Comput.* 3, 2, Article 7 (February 2022), 29 pages.
https://doi.org/10.1145/3505181

1 INTRODUCTION

With the steady advances in quantum hardware and volume [32], quantum computing is well on track to become widely adopted in science and technology in the near future. One of the core challenges to enable its use is the availability of a flexible and reliable quantum compiler, which can translate any target quantum circuit into a circuit that can be implemented on real hardware with gate set, connectivity, and length limitations.

Since the celebrated Solovay-Kitaev theorem [19, 52], quantum compiling has been a rich research area. Works have investigated how to efficiently map different gates into canonical (universal) gate sets up to an arbitrary accuracy [3, 35, 42, 48, 60–62, 74], or how to "place" the target circuit onto the real connectivity-limited hardware [9–11, 16, 29, 43, 46, 49, 53, 59, 67–69, 76, 77].

A more holistic strategy in quantum compiling has been the construction of universal parametric circuits to serve as templates to compile any target circuit. This line of research, which we will call decomposition-based, focuses primarily on templates based on the versatile CNOT+rotation gate set [4, 13, 21, 47, 63, 65, 70]. These works unveiled fundamental lower bounds on the number

This project has received funding from the Disruptive Technologies Innovation Fund (DTIF), by Enterprise Ireland, under project number DTIF2019-090.
The authors performed part of the work while at IBM Quantum, IBM Research Europe.
Authors' addresses: L. Madden, University of Colorado Boulder, 425 UCB, Boulder, Colorado CO 80309; email: Liam.Madden@colorado.edu; A. Simonetto, UMA, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau 91120, France; email: andrea.simonetto@ensta-paris.fr.

2643-6817/2022/02-ART7
https://doi.org/10.1145/3505181

of CNOTs that almost all target circuits require in order to be compiled in such a gate set and delivered a constructive method for doing so, the quantum Shannon decomposition (QSD), which was only a factor of two off from the lower bound of efficiency. This research is formalized mainly in the language of Lie theory as a recursive sequence of Cartan decompositions. Despite the constructive and sound theory, the QSD decomposition does not have the flexibility of trading off precision and length.

The quantum compiling problem in its essence can be cast as an optimization problem in the space of unitary matrices. Here one needs to find a unitary matrix that can be realized in hardware (with various constraints, e.g., gate set and connectivity) that is the "closest" to a target unitary (i.e., the circuit that one wants to realize, or compile). Here "closest" is intended with respect to a pertinent metric. On the one hand, this optimization problem could encompass the whole quantum compiling research; on the other hand, it is a very difficult mathematical problem and even for a small number of qubits cannot be stored in memory. Recently, a series of articles [15, 33], have revised this optimization-based compiling approach with some simplifying assumptions and heuristics, and have introduced the idea of computing the cost and its gradients in a quantum-assisted way. With the same optimization lens, but with other classical heuristics, the works [72, 73] have looked at hierarchical compilations, whereby one need not compile a unitary directly to a two-qubit gate circuit, but can instead start with higher-qubit gates, and then down recursively to the two-qubit gate target. Finally, the recent works [57, 58] arrange the CNOTs to explicitly decouple the qubits one at a time using a particular cost function. Moreover, they provide numerical evidence that their approach can compile arbitrary target circuits very close to the lower bound on the number of CNOTs.

In this article, we aim at analyzing the optimization approach in more depth and offering some sound evidence on the justification of critical assumptions and solution methods. Further, we aim at helping to bridge the gap between Cartan decomposition-based research and optimization-based compiling. Our approach consists in formulating best approximate compiling problems, which trade-off exact compilation with constraint violation. In particular, we offer the following contributions.

— We start by analyzing the quantum compiling mathematical problem in the CNOT+rotation gate set and show how to construct versatile "CNOT units," i.e., elementary building blocks, that can be used to parametrize any circuit of a given number of qubits;
— We show that optimizing over the parametrized circuit consists of optimizing the structure (i.e., where to place the CNOT units) and optimizing the rotation angles of the rotation gates. We show that the former is largely unimportant once past the so-called surjectivity bound (even when imposing hardware constraints), while the latter is easy from an optimization perspective, by using e.g., Nesterov's accelerated gradient descent [5, 51];
— With the intention to further compress the compiled circuit, allowing for approximation errors in terms of gate fidelity, we propose a novel regularization based on group LASSO [66], which (among other things) can reduce the length of the QSD down to the CNOT theoretical lower bound without affecting fidelity noticeably (i.e., a factor of two compression);
— Various numerical results support our findings. In particular, we showcase how to use our approach to discover new decompositions in the CNOT+rotation gate set of special gates (e.g., the Toffoli gate) and how to use the compression mechanism as an extension of any compilation code available (e.g., Qiskit transpile [56]) that can trade-off accuracy for circuit *length*, where length is the number of CNOTs.

While discussing the main contributions, the article focuses on three complementary goals,

- **[G1]** Approximate quantum compiling for random unitary matrices: here the goal is to derive results in terms of the number of CNOTs to use to compile any given random unitary matrix, with connectivity constraints;
- **[G2]** Approximate quantum compiling for special gates: here the goal is to derive results in terms of the number of CNOTs to use to compile special gates;
- **[G3]** Approximate quantum compiling for circuit compression: here the goal is to derive results that allow for compression of circuits, freeing the possibility to have inexact compilation.

Organization. This article is organized as follows. In Section 2, we report the mathematical and physical preliminaries to our algorithmic development. In Section 3, we formalize the approximate quantum compiling problem as a mathematical program. In Section 4, we devise a programmable unit (the two-qubit CNOT unit) with which we can build any circuit. In Section 5, we discuss the property of the mathematical program introduced in Section 3 when specified for the parametric circuit presented in Section 4.

From Section 6, we focus on particular layout patterns, namely sequential, spin, and Cartan, and we present their properties. In Section 7, we discuss the use of gradient descent to optimize the rotation angles once the layout is fixed, and we showcase numerical results in Sections 7.2–7.3. Section 8 discusses our proposed compression strategy to trade-off accuracy and length, both theoretically and numerically. We then conclude in Section 9.

A version of the code used in this article is made available in the Qiskit AQC synthesis package (`https://qiskit.org/documentation/apidoc/synthesis_aqc.html`).

2 PRELIMINARIES

We work with n-qubit quantum circuits, which are represented either by a collection of ordered gate operations, or by a d by d unitary matrix with $d = 2^n$. The class of unitary matrices of dimension $d \times d$ together with the operation of matrix multiplication have an important group structure [36], denoted as the unitary group $\mathrm{U}(d)$. In particular, the group is a Lie group of dimension d^2 as a real manifold; we let $\mathfrak{u}(d)$ be its Lie algebra, which consists of anti-Hermitian matrices. Unitary matrices with determinant equal to 1 are called special unitary matrices. Special unitary matrices of dimension $d \times d$ together with the operation of matrix multiplication form the the special unitary group of degree d: $\mathrm{SU}(d)$, which is a Lie group of dimension $d^2 - 1$ as a real manifold. We let $\mathfrak{su}(d)$ denote its Lie algebra, which consists of traceless anti-Hermitian matrices.

A useful property of the determinant $\det(\cdot)$ of any squared d by d matrix A and scalar c is that $\det(cA) = c^d A$. Hence if $U \in \mathrm{U}(d)$ then $U/\det(U)^{1/d} \in \mathrm{SU}(d)$. Since scalars (e.g., global phases in quantum computing) are easy to implement on a quantum computer (and in fact unimportant), we normalize unitary matrices as above, and so we only need to know how to "work with" special unitary matrices. We remark that if $U \in \mathrm{SU}(d)$, then adding any global phase multiple of $2\pi/d$ does not alter the matrix determinant, i.e., $e^{i\frac{2\pi m}{d}} U \in \mathrm{SU}(d)$ as well for any integer $m \in \mathbb{Z}$. The equivalence classes from this relation form the projective special unitary group, $\mathrm{PSU}(d)$, which is isomorphic to the projective unitary group, $\mathrm{PU}(d) := \mathrm{U}(d)/\mathrm{U}(1)$. Thus, compiling a matrix from $\mathrm{SU}(d)$ also provides a compilation for its $d - 1$ equivalent matrices in $\mathrm{PSU}(d)$.

A single-qubit gate on the jth qubit is a unitary matrix of the form $I_{2^{j-1}} \otimes u \otimes I_{2^{n-j}}$ where $u \in \mathrm{U}(2)$, I_q is the identity matrix of dimension q, and $\otimes$ represents the Kronecker product. An important set of matrices in $\mathrm{U}(2)$ is the set of Pauli matrices, which we denote with their usual notation as X, Y, Z, which are unitary, Hermitian, traceless, and have determinant equal to -1. From the Pauli matrices, one obtains the rotation matrices $R_x(\theta), R_y(\theta), R_z(\theta)$ by matrix exponentiation. The rotation matrices are special unitary. An important fact that we use extensively in this article is that any $u \in \mathrm{SU}(2)$ can be written as a product of any three rotation matrices with no two consecutive the same [52].

Among multiple-qubit gates, we focus on the two-qubit CNOT gate, or controlled-X gate, with control qubit j and target qubit k, which is the matrix

$$\mathrm{CNOT}_{jk} = I_{2^{j-1}} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes I_{2^{n-j}} + \begin{cases} I_{2^{j-1}} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes I_{2^{k-j-1}} \otimes X \otimes I_{2^{n-k}} & \text{if } j < k \\ I_{2^{k-1}} \otimes X \otimes I_{2^{j-k-1}} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes I_{2^{n-j}} & \text{if } k < j \end{cases}.$$

For example, for $n = 2$, we have

$$\mathrm{CNOT}_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad \mathrm{CNOT}_{21} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Note that for $n = 2$, CNOT has determinant -1, and so we have to normalize it. On the other hand, for $n > 2$, CNOT has determinant 1.

3 APPROXIMATE QUANTUM COMPILING AS MATHEMATICAL OPTIMIZATION

We are interested in compiling a quantum circuit, which we formalize as finding the "best" circuit representation in terms of an ordered gate sequence of a target unitary matrix $U \in \mathrm{U}(d)$, with some additional hardware constraints. In particular, we look at representations that could be constrained in terms of hardware connectivity, as well as circuit length, and we choose a gate basis in terms of CNOT and rotation gates. The latter choice is motivated by an implementation in the Qiskit software package [56]. We recall that the combination of CNOT and rotation gates is universal in $\mathrm{SU}(d)$ and therefore it does not limit compilation [52].

To properly define what we mean by "best" circuit representation, we define the metric as the Frobenius norm between the unitary matrix of the compiled circuit V and the target unitary matrix U, i.e., $\|V - U\|_{\mathrm{F}}$. This choice is motivated by mathematical programming considerations, and it is related to other formulations that appear in the literature (see Remark 1).

We are now ready to formalize the approximate quantum compiling problem as follows.

Given a target special unitary matrix $U \in SU(2^n)$ and a set of constraints, in terms of connectivity and length, find the closest special unitary matrix $V \in \mathcal{V} \subseteq SU(2^n)$, where $\mathcal{V}$ represents the set of special unitary matrices that can be realized with rotations and CNOT gates alone and satisfy both connectivity and length constraints, by solving the following mathematical program:

$$\textbf{(AQCP)} \qquad \min_{V \in \mathcal{V} \subseteq SU(2^n)} f(V) := \frac{1}{2} \|V - U\|_{\mathrm{F}}^2. \tag{1}$$

Note that the cost function can be equivalently written

$$\begin{aligned} \frac{1}{2}\|V - U\|_F^2 &= \frac{1}{2}\mathrm{Tr}[U^\dagger U - U^\dagger V - V^\dagger U + V^\dagger V] = \frac{1}{2}\mathrm{Tr}[2I] - \frac{1}{2}\mathrm{Tr}[U^\dagger V] - \frac{1}{2}\mathrm{Tr}[V^\dagger U] \\ &= d - \mathbb{R}\mathrm{e}\,\mathrm{Tr}[U^\dagger V], \end{aligned} \tag{2}$$

where we used that U and V are unitary matrices.

We call Equation (1) the **approximate quantum compiling (master) problem** (**AQCP**). A solution of the problem is an optimal V indicated as V^*, along with an ordered set of gate operations that respect the constraints.

If V^* is such that the cost is null, then we say that the compilation is exact, otherwise it is approximate and the approximation error is computed by the cost $\frac{1}{2}\|V^* - U\|_{\mathrm{F}}^2$. Without further specifications (and simplifications), Problem (1) is intractable but for very small circuits. How to efficiently solve Equation (1) is our aim.

Remark 1. In [33], a different metric was used, namely the Hilbert-Schmidt test defined as

$$C_{HST}(U,V) = 1 - \frac{1}{d^2}|\mathrm{Tr}[V^\dagger U]|^2 = \frac{d+1}{d}(1 - \bar{F}(U,V)),$$

where $\bar{F}(U,V)$ is the fidelity averaged over the Haar distribution. Given that $f(V) = \frac{1}{2}\|V - U\|_{\mathrm{F}}^2 = d - \mathbb{R}\mathrm{e}\mathrm{Tr}[V^\dagger U]$, then $\mathbb{R}\mathrm{e}\mathrm{Tr}[V^\dagger U] = d - f(V)$. Hence,

$$\begin{aligned}\bar{F}(U,V) = 1 - \frac{d}{d+1}C_{HST}(U,V) &= 1 - \frac{d}{d+1} + \frac{1}{d(d+1)}((\mathbb{R}\mathrm{e}\mathrm{Tr}[V^\dagger U])^2 + (\mathbb{I}\mathrm{m}\mathrm{Tr}[V^\dagger U])^2) \\ &\geq 1 - \frac{d}{d+1} + \frac{1}{d(d+1)}(d - f(V))^2 =: \bar{F}_{\mathrm{F}}(U,V),\end{aligned}$$

where we have defined the new quantity $\bar{F}_{\mathrm{F}}(U,V)$ as the Frobenius fidelity, which is always lower than the $\bar{F}(U,V)$. By minorization arguments, minimizing $f(V)$ has the effect of maximizing the Frobenius fidelity.

The stepping stones of our work are the articles [15, 33]. There, the AQCP is approached by a bi-level technique. Since V needs to be sought in the space of matrices that can be realized with rotation and CNOT gates, one can solve for V by interleaving an optimization on the structure (at fixed length), i.e., which gate needs to be implemented where, and an optimization on the rotation angles. The first problem (deciding the structure) is combinatorial and non-convex, and [15, 33] propose a meta-heuristic based on simulated-annealing and compactifications, while the second problem (deciding the rotation angles) is continuous and non-convex, and [15, 33] include a gradient descent algorithm. [15, 33] also offer an algorithm to fix the CNOT structure, more or less arbitrarily, and optimize only for rotation angles, which is more efficient in terms of computation time, but less optimal.

Despite the encouraging results, key challenges in these works remain. First, optimizing the structure via simulated-annealing is far from optimal and it can be very time consuming. But most importantly it is physics-agnostic, therefore one may wonder if one could do better by using physics.

Second, fixing the structure, as is done in [33], is by no means universal nor does it mirror the hardware connectivity; a better structure could be a sequential structure (see Figure 1).

Third, the gradient descent algorithm used in [33] seems to work surprisingly well, despite the non-convexity of the problem at hand, and one may wonder why this is so, which is a non-trivial question.

The rest of this article answers these questions by using a pertinent parametrization of the matrix V in terms of CNOTs and rotation gates.

4 A PROGRAMMABLE TWO-QUBIT GATE AND THE CNOT UNIT

We start by defining a flexible two-qubit gate, which is parametrized by four rotation angles, and which will be the building block of the parametric circuit.

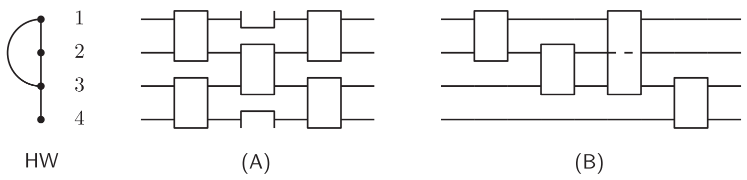


Fig. 1. Possible two-qubit gate units structures. HW depicts the hardware connectivity of a four-qubit quantum computer. (A) the structure, or ansatz, proposed in [33], where each block represents a CNOT gate with rotation gates before and after, is easily implementable but may miss some hardware connections or introduce some that are not there; (B) a possible sequential structure, which alternates among all the possible two-qubit blocks and could capture hardware connectivity better.

Let us focus for now on basis set constraints alone. Given the target special unitary $U \in \mathrm{SU}(d)$, compiling means to find a product of matrices from the basis set (as said CNOT+rotations) that either approximates or equals U. We define the "length" of the compilation as the number of CNOTs in the product. In 2004, Shende, Markov, and Bullock derived a universal lower bound on the length for exact compilation in this setting [65]. The bound specifies a *sufficient* minimal number of CNOTs for exact compilation, for *all* the matrices in $\mathrm{SU}(d)$. And, while a particular special unitary matrix may be exactly compiled with a smaller length than the universal lower bound (the bound is not *necessary*), there exists a special unitary matrix that cannot (the bound is sufficient and necessary for at least one special unitary matrix). Moreover, the set of special unitary matrices that can be compiled with a smaller length than the universal lower bound has measure zero in $\mathrm{SU}(d)$.

We summarize the proof of this fact, as it motivates our parametrization ideas.

Lemma 1 ([65], Prop. III.1). *The set of special unitary matrices $U \in SU(d)$, $d = 2^n$, that do not need at least $\lceil \frac{1}{4}(4^n - 3n - 1) \rceil$ CNOTs to be exactly compiled has measure zero in $SU(d)$.*

Proof. First, since each rotation gate has 1 real parameter, while $\mathrm{SU}(d)$ has real dimension d^2-1, Sard's theorem [38] can be used to show that the set of special unitary matrices that do not need $d^2 - 1$ rotation gates to be exactly compiled has measure zero in $\mathrm{SU}(d)$ [65, Lem. II.2]. Thus, if a product of matrices from the basis set can be reduced to a product with less than $d^2 - 1$ rotation gates, then it is in the measure 0 set. In particular, more than 3 consecutive rotation gates on the same qubit in a product can be reduced to only 3 rotation gates. So, without CNOTs, any product can be reduced to one with only $3n < d^2 - 1$ rotation gates. Furthermore, R_z commutes with the control of CNOTs and R_x commutes with the target of CNOTs. Thus, whenever 3 rotation gates are applied after a CNOT on either the control or target qubit, we can rewrite them as 3 rotation gates such that one of the rotation gates commutes with the CNOT. Thus, we can reduce any product of matrices from the basis set to a product with 3 rotation gates applied to each qubit, followed by CNOTs, each followed by only 4 rotation gates. Thus, a product with L CNOTs can be reduced to a product with $4L + 3n$ rotation gates. So, the set of special unitary matrices that do not need at least $\lceil \frac{1}{4}(4^n - 3n - 1) \rceil$ CNOTs to be exactly compiled has measure zero in $\mathrm{SU}(d)$ [65, Prop. III.1]. □

Therefore, for an arbitrary n-qubit circuit, one would need at least $\Omega(4^n)$ CNOT gates for exact compilation. Henceforth, we refer to the bound $\lceil \frac{1}{4}(4^n - 3n - 1) \rceil$ as the **theoretical lower bound** (**TLB**). Figure 2 gives a glimpse of how the reduction in terms of CNOTs and rotations can be carried out for a 3-qubit circuit.

While [65] only proved a lower bound, their reasoning motivates a flexible parametric circuit construction. Since every product of matrices from the basis set can be reduced to a product of

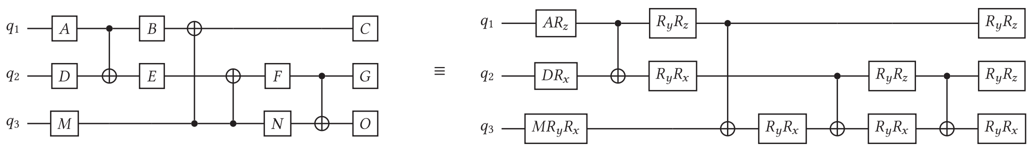


Fig. 2. An example of an equivalent quantum circuit with CNOT and rotations gates, all the capital letter gates, e.g., A, are single-qubit unitaries that can be compiled via three rotation gates up to an irrelevant global phase. Note that the R_y next to M results from flipping the two upward-facing CNOTs and has angle $-\pi/2$.

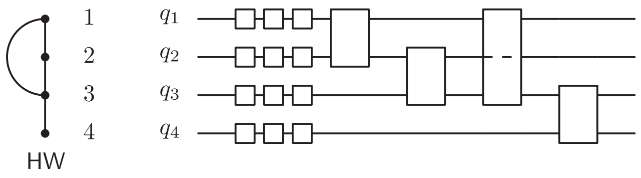
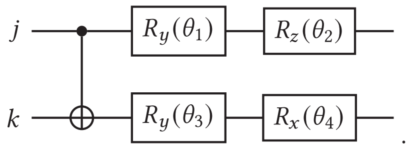


Fig. 3. Decomposition of a realizable quantum circuit in terms of $V_{\mathrm{ct}}(\boldsymbol{\theta})$. The first three one-qubit gates for all qubits are rotation gates, R_z, R_y, and $\mathrm{R_z}$, while the two-qubit blocks represents CNOT units as specified. Here, $L = 4$; hence, there are 28 rotation angles variables. Moreover, $\mathrm{ct} = (1 \to 2, 2 \to 3, 1 \to 3, 3 \to 4)$, so the hardware connectivity is satisfied.

the form described in the previous paragraph (see also Figure 2), we will consider products of that form with different sequences of CNOTs. Let $\mathrm{CU}_{j\to k}(\theta_1, \theta_2, \theta_3, \theta_4)$ be the matrix represented by



We call this a "CNOT unit" and we use it as a programmable two-qubit block in the parametric circuit. Define

$$J(L) = \{(j_1 \to k_1, \ldots, j_L \to k_L) \mid j_m < k_m \in \{1, \ldots, n\} \text{ for all} m \in \{1, \ldots, L\}\},$$

as the set of L-long lists of pairs of control-target indices with the control index smaller than the target index. The latter constraint is without loss of generality since a CNOT can be flipped with y-rotations ($R_y(\pi/2)$ on the left of the control and right of the target, and $R_y(-\pi/2)$ on the right of the control and left of the target). The set $J(L)$ represents all the possible CNOT unit sequences that one can have of length L. In Figure 2, we have one element of $J(4)$ as $(1 \to 2, 1 \to 3, 2 \to 3, 2 \to 3)$.

The cardinality of $J(L)$ can be shown to be $|J(L)| = (n(n-1)/2)^L$. Given an element $\mathrm{ct} \in J(L)$ (that is, an L-long list of pairs defining the location of our CNOT units, for example in Figure 2 $\mathrm{ct} = (1 \to 2, 1 \to 3, 2 \to 3, 2 \to 3)$), we define $\mathrm{ct}(i)$ as the pair at position i, and we let $V_{\mathrm{ct}} : [0, 2\pi)^{3n+4L} \to \mathrm{SU}(2^n)$ be the following circuit

$$V_{\mathrm{ct}}(\boldsymbol{\theta}) = \mathrm{CU}_{\mathrm{ct}(L)}(\theta_{3n+4L-3}, \ldots, \theta_{3n+4L}) \cdots \mathrm{CU}_{\mathrm{ct}(1)}(\theta_{3n+1}, \ldots, \theta_{3n+4})$$
$$[R_z(\theta_1)R_y(\theta_2)R_z(\theta_3)] \otimes \cdots \otimes [R_z(\theta_{3n-2})R_y(\theta_{3n-1})R_z(\theta_{3n})],$$

where we have collected all the angles $(\theta_1, \ldots, \theta_{3n+4L})$ into the vector $\boldsymbol{\theta}$. The circuit V_{ct} corresponds to the reduced product in the lower bound proof and will be the main object of our study: it will be the circuit blueprint for any circuit realizable in hardware (see also Figure 3).

Remark 2. A useful circuit decomposition is the QSD, proposed in 2006 by Shende, Bullock, and Markov, which uses $\frac{23}{48}4^n - \frac{3}{2}2^n + \frac{4}{3}$ CNOTs to decompose any n-qubit circuit [63]. This number of CNOTs is only twice the TLB. The resulting circuit is formalized as a sequence of recursive Cartan decompositions in [21], and Theorems 4, 8, and 12 in [63] can be recursively used to find an explicit ct corresponding to the QSD. For $n = 3$, $\mathrm{ct} = (1 \to 2, 1 \to 2, 1 \to 2, 2 \to 3, 1 \to 3, 2 \to 3, 1 \to 2, 1 \to 2, 1 \to 2, 2 \to 3, 1 \to 3, 2 \to 3, 1 \to 3, 1 \to 2, 1 \to 2, 1 \to 2, 2 \to 3, 1 \to 3, 2 \to 3, 1 \to 2, 1 \to 2, 1 \to 2)$ [21, Figure 3].

At this point, we can already reformulate the approximate quantum compiling problem (1), in the equivalent form

$$\textbf{(AQCP-CT)} \qquad \min_{L\in[1,\ldots,\bar{L}],\boldsymbol{\theta}\in[0,2\pi)^{3n+4L},\mathrm{ct}\in C(L)} \quad f_{\mathrm{ct}}(\boldsymbol{\theta}) := \frac{1}{2}\,\|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_{\mathrm{F}}^2, \tag{3}$$

where the set $C(L)$ represents the set of L-long realizable lists in hardware (for connectivity limitations), and $\bar{L}$ is the length limit.

Problem (3) is by no means easier than the original Problem (1). However, it is instructive to understand its properties, and this will help us devise better solution strategies. For instance, if we were to drop hardware constraints and adopt the QSD strategy of Remark 2, then Problem (3) could simplify into

$$\textbf{(AQCP-QSD)} \qquad \min_{L=\frac{23}{48}4^n-\frac{3}{2}2^n+\frac{4}{3},\boldsymbol{\theta}\in[0,2\pi)^{3n+4L},\mathrm{ct}=\mathrm{QSD}(n)} \quad f_{\mathrm{ct}}(\boldsymbol{\theta}) := \frac{1}{2}\,\|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_{\mathrm{F}}^2, \tag{4}$$

which is a non-convex (but continuous) optimization problem in the $\boldsymbol{\theta}$ variables. In fact, since the structure is fixed, the only free parameters are the rotation angles. But on the other hand, can we do better in terms of number of CNOTs and in terms of incorporating the hardware constraints?

5 THE OPTIMIZATION LANDSCAPE

We start by deriving some useful properties of the cost function $f_{\mathrm{ct}}(\boldsymbol{\theta}) := \frac{1}{2}\|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_{\mathrm{F}}^2$ given the target unitary U and a fixed structure. In particular, we will look at V_{ct} and its properties of differentiability, as well as the first and second order derivatives of f_{ct}, which are important for optimization purposes.

First, we would like to prove that $V_{\mathrm{ct}}(\boldsymbol{\theta})$ is a smooth mapping in the rotation angles $\boldsymbol{\theta}$. To do so, we apply the partial derivative operator, $\frac{\partial}{\partial\theta_k}$, to V_{ct}, with respect to an arbitrary angle θ_k, where k is an index in $[1, 3n + 4L]$. We further let R_{g_k} be the rotation gate associated to that angle and σ_{g_k} be the respective Pauli operator (g can be x, y, and z depending on which type of rotation gate the angle refers to). With this notation, $\frac{d}{d\theta_k}R_{g_k}(\theta_k) = -\frac{i}{2}\sigma_{g_k}R_{g_k}(\theta_k)$, and $\frac{d^2}{d\theta_k^2}R_{g_k}(\theta_k) = -\frac{1}{4}R_{g_k}(\theta_k)$. Thus, if a sequence of partial derivative operators are applied to V_{ct}, to compute its gradient, or Hessian, or higher order derivative, the result is a complex constant times V_{ct} with Pauli gates inserted at specific locations. Thus, we have proved the following theorem.

Theorem 1. *The circuit $V_{\mathrm{ct}}(\boldsymbol{\theta})$ is infinitely differentiable with respect to the rotation angles $\boldsymbol{\theta}$.*

Next, we look at the cost $f_{\mathrm{ct}}(\boldsymbol{\theta})$. Using Equation (2), we get

$$\frac{\partial}{\partial\theta_k}\frac{1}{2}\,\|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_F^2 = -\mathrm{Re}\,\mathrm{Tr}\left[\frac{\partial}{\partial\theta_k}V_{\mathrm{ct}}(\boldsymbol{\theta})^\dagger U\right],$$

and

$$\frac{\partial^2}{\partial\theta_k\partial\theta_\ell}\frac{1}{2}\,\|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_F^2 = -\mathrm{Re}\,\mathrm{Tr}\left[\frac{\partial^2}{\partial\theta_k\partial\theta_\ell}V_{\mathrm{ct}}(\boldsymbol{\theta})^\dagger U\right]. \tag{5}$$

In particular, by the fact that $\frac{d^2}{d\theta_k^2} R_{g_k}(\theta_k) = -\frac{1}{4} R_{g_k}(\theta_k)$, for the diagonal elements

$$\frac{\partial^2}{\partial \theta_k^2} \frac{1}{2} \|V_{\text{ct}}(\boldsymbol{\theta}) - U\|_F^2 = \frac{1}{4} \mathbb{R}\text{e}\, \text{Tr}[V_{\text{ct}}(\boldsymbol{\theta})^\dagger U] = \frac{d}{4} - \frac{1}{8} \|V_{\text{ct}}(\boldsymbol{\theta}) - U\|_F^2 . \tag{6}$$

Now we are ready to show that $f_{\text{ct}}(\boldsymbol{\theta})$ is strongly smooth in the optimization sense (i.e., its gradient is Lipschitz continuous).

THEOREM 2. *For all $\boldsymbol{\theta}$, the Hessian of $f_{\text{ct}}(\cdot) = \frac{1}{2}\|V_{\text{ct}}(\cdot) - U\|_F^2$ evaluated at $\boldsymbol{\theta}$ has spectrum in $[-(3n + 4L - 3/4)d, (3n + 4L - 3/4)d]$.*

PROOF. First of all, the Hessian is real and symmetric, so its eigenvalues lie on the real line. Then, note that for all unitary matrices, $W \in \text{U}(d)$: $\|W\|_{\text{F}}^2 = d$, and the farthest unitary matrix from any W is $-W$. Thus, $\mathbb{R}\text{e}\, \text{Tr}[\cdot^\dagger \cdot] : \text{U}(d)^2 \to [-d, d]$. Hence, using Equation (5), we can show that each off-diagonal element of the $(3n + 4L)$ by $(3n + 4L)$ Hessian matrix is bounded in absolute value by d. As for the diagonal elements, Equation (6) says that they are bounded below by $-d/4$ and above by $d/4$.

With this in place, we can use Gershgorin's disc theorem [27, Theorem 6.1.1.] with centers in $[-d/4, d/4]$ and radii in $[0, (3n + 4L - 1)d]$ to prove the claim. □

Theorem 2 says that $f_{\text{ct}}(\boldsymbol{\theta})$ is a strongly smooth function (since the spectrum of the Hessian is uniformly bounded), which implies fast convergence to a stationary point (i.e., points for which the gradient vanishes) for gradient descent. In addition, even though $f_{\text{ct}}(\boldsymbol{\theta})$ is non-convex, gradient descent with random initialization [37], perturbed gradient descent [30], and perturbed Nesterov's method [31] all converge to second-order stationary points under strong smoothness. Second-order stationary points are stationary points where the Hessian is positive semi-definite, and therefore are local minima. So, applying these methods to $f_{\text{ct}}(\boldsymbol{\theta})$, we will be sure to reach at least a local minimum.

In some cases, such as in non-convex low rank problems, most second-order stationary points are, in fact, global minima [24]. Even though this is not the case for f_{ct}, we do not need to converge to a global minimum necessarily. We would be content finding a $\boldsymbol{\theta}^*$ such that $V_{\text{ct}}(\boldsymbol{\theta}^*)$ is a global minimum up to a global phase transformation.

To this aim, we now present supporting facts that lead us to conjecture that, under certain conditions, we can easily find global minima up to a global phase transformation.

First, we report a technical lemma.

LEMMA 2. *Indicate with $^{\perp_{\mathbb{R}\text{e}\langle\cdot,\cdot\rangle}}$ the orthogonal complement with respect to the $\mathbb{R}\text{e}\text{Tr}[(\cdot)^\dagger(\cdot)]$ operation. The orthogonal complement $\mathfrak{u}(d)^{\perp_{\mathbb{R}\text{e}\langle\cdot,\cdot\rangle}}$ is the set of Hermitian matrices and $\mathfrak{su}(d)^{\perp_{\mathbb{R}\text{e}\langle\cdot,\cdot\rangle}} = \mathfrak{u}(d)^{\perp_{\mathbb{R}\text{e}\langle\cdot,\cdot\rangle}} + \text{span}_{\mathbb{R}}\{iI\}$.*

PROOF. Recall that $\mathfrak{u}(d)$ consists of anti-Hermitian matrices and $\mathfrak{su}(d)$ consists of traceless anti-Hermitian matrices. The orthogonal complement $\mathfrak{u}(d)^{\perp_{\mathbb{R}\text{e}\langle\cdot,\cdot\rangle}}$ is the set of all the matrices H for which $\mathbb{R}\text{e}\text{Tr}[H^\dagger U] = 0$, for all U anti-Hermitian. By direct calculation, indicating with h_{ij} the element i, j of matrix H, and with $u_{i,j}$ the one of matrix U, as well as $\bar{h}_{i,j}$ the conjugate of $h_{i,j}$, then,

$$\mathbb{R}\text{e}\text{Tr}[H^\dagger U] = \mathbb{R}\text{e} \left[\sum_i \sum_j \bar{h}_{i,j} u_{i,j} \right].$$

If the above has to be 0 for all anti-Hermitian matrices U, then H has to be Hermitian by $u_{j,i} = -\bar{u}_{i,j}$ (in particular, the diagonal of U is only imaginary). On the other hand, if H is Hermitian (therefore its diagonal is only real), then,

$$\mathbb{R}\mathrm{e}\mathrm{Tr}[H^\dagger U] = \underbrace{\mathbb{R}\mathrm{e}\Big[\sum_{j>i} \bar{h}_{i,j}u_{i,j} + \sum_{j<i} h_{i,j}\bar{u}_{i,j}\Big]}_{=0} + \underbrace{\mathbb{R}\mathrm{e}\Big[\sum_{i=j} \bar{h}_{i,j}u_{i,j}\Big]}_{=0} = 0.$$

Hence, the orthogonal complement $\mathfrak{u}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}}$ is the set of Hermitian matrices.

In addition, $\mathfrak{su}(d)$ consists of traceless anti-Hermitian matrices, so its orthogonal complement consists not only of Hermitian matrices, but also of any matrix of the form $H + ciI$, where H is Hermitian and $c \in \mathbb{R}$. Thus, the thesis follows. □

With the Lemma in place, we are ready for another intermediate result that pertains to stationary points.

Theorem 3. *Let $U \in SU(d)$ be a special unitary matrix, and $V(\cdot) : \mathbb{R}^p \to SU(d)$ be a function that is infinitely differentiable, surjective, and of constant rank. For all $U \in SU(d)$, a parameter value $\boldsymbol{\theta} \in [0, 2\pi)^p$ is a stationary point of the cost $f(\cdot) = \frac{1}{2}\|V(\cdot) - U\|_F^2$ if and only if $V(\boldsymbol{\theta})^\dagger U$ has eigenvalues in $\{e^{i\alpha}, -e^{-i\alpha}\}$ for some $\alpha \in [0, 2\pi)$.*

Proof. Assume the hypotheses. We have the following line of implications:

$$\begin{aligned}
\boldsymbol{\theta} \text{ is a stationary point of } f \quad &\overset{1}{\iff} \mathbb{R}\mathrm{e}\ \mathrm{Tr}\left[\frac{\partial}{\partial\theta_k} V(\boldsymbol{\theta})^\dagger U\right] = 0 \text{ for all } k \in [p] \\
&\overset{2}{\iff} U \in \mathrm{span}_{\mathbb{R}}\left\{\frac{\partial}{\partial\theta_k} V(\boldsymbol{\theta})\right\}^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \\
&\qquad\qquad \overset{3}{=} \left[\mathcal{T}_{V(\boldsymbol{\theta})}\mathrm{SU}(d)\right]^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \\
&\qquad\qquad \overset{4}{=} [V(\boldsymbol{\theta})\mathfrak{su}(d)]^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \\
&\qquad\qquad \overset{5}{=} V(\boldsymbol{\theta})\mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \\
&\iff V(\boldsymbol{\theta})^\dagger U \in \mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}}.
\end{aligned}$$

Step 1 follows from the definition of a stationary point (a point where the gradient vanishes) and our derivation of the gradient for $f(\cdot) = \frac{1}{2}\|V(\cdot) - U\|_F^2$. Step 2 follows from the definition of an orthogonal complement and the linearity of the trace (i.e., U is in the span of the orthogonal complement of the derivative of V if and only if the gradient is 0). Step 3 follows from the global rank theorem [38, Thm 4.14] under infinite differentiability, surjectivity, and constant rank assumptions, where we use $\mathcal{T}$ to denote the tangent space (which appears since we are taking the derivative of $V(\boldsymbol{\theta})$). Concretely, given $W \in \mathrm{SU}(d)$, $\mathcal{T}_W\mathrm{SU}(d) := \{\gamma'(0) \mid \gamma : \mathbb{R} \to \mathrm{SU}(d) \text{ smooth with } \gamma(0) = W\}$. For Step 4, a little more care has to be put. First, it can be shown, via the left or right group action, that $\mathcal{T}_W\mathrm{SU}(d)$ is isomorphic to $\mathcal{T}_I\mathrm{SU}(d) = \mathfrak{su}(d)$. We want now to show that $\mathcal{T}_W\mathrm{SU}(d) = W\mathfrak{su}(d)$. Since they are isomorphic, we only have to show one direction. Let $A \in \mathfrak{su}(d)$. We want to show $WA \in \mathcal{T}_W\mathrm{SU}(d)$. Toward this end, define $\gamma(t) = W\exp(tA)$. Note $\gamma(0) = W$, and $\gamma'(0) = WA$. Furthermore, $\gamma(t) \in \mathrm{SU}(d)$ (which is easy to see by direct computations[1]). Thus, $WA = \gamma'(0) \in \mathcal{T}_W\mathrm{SU}(d)$ and so $\mathcal{T}_W\mathrm{SU}(d) = W\mathfrak{su}(d)$, proving Step 4. Step 5 follows via properties of the Hermitian

[1]We have $\gamma(t)^\dagger\gamma(t) = \exp(tA)^\dagger W^\dagger W\exp(tA) = \exp(tA)^\dagger\exp(tA) = \exp(tA^\dagger)\exp(tA) = \exp(-tA)\exp(tA) = \exp(-tA + tA) = I$ and $\det(\gamma(t)) = \det(W\exp(tA) = \det(W)\det(\exp(tA)) = \det(\exp(tA)) = \exp(t\mathrm{Tr}(A)) = \exp(0) = 1$.

transpose. Explicitly, for a set $\mathcal{U}$,

$$\begin{aligned}[W\mathcal{U}]^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} &= \{Y \mid \mathbb{R}\mathrm{e}\,\mathrm{Tr}[Y^\dagger WA] = 0 \text{ for all } A \in \mathcal{U}\}\\ &= \{Y \mid \mathbb{R}\mathrm{e}\,\mathrm{Tr}[(W^\dagger Y)^\dagger A] = 0 \text{ for all } A \in \mathcal{U}\}\\ &= W\{W^\dagger Y \mid \mathbb{R}\mathrm{e}\,\mathrm{Tr}[(W^\dagger Y)^\dagger A] = 0 \text{ for all } A \in \mathcal{U}\}\\ &= W\mathcal{U}^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}}.\end{aligned}$$

Now all that is left is to determine the contents of $\mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \cap \mathrm{U}(d)$. We have from Lemma 2 that $\mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}}$ is the set of matrices that equal $A + ciI$ for some Hermitian A and some $c \in \mathbb{R}$. Furthermore, Hermitian matrices are precisely those that equal $Q^\dagger DQ$ for some unitary Q and diagonal D with real entries. So, $\mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}}$ is the set of matrices that are unitarily diagonalizable with eigenvalues in $\{\lambda + ci \mid \lambda \in \mathbb{R}\}$ for some $c \in \mathbb{R}$. But, unitary matrices are precisely the matrices that are unitarily diagonalizable with eigenvalues in $\mathrm{U}(1)$. Thus, $\mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \cap \mathrm{U}(d)$ is the set of matrices that are unitarily diagonalizable with eigenvalues in $\{e^{i\alpha}, -e^{-i\alpha}\}$ for some α, since $\{\lambda + ci \mid \lambda \in \mathbb{R}\} \cap \mathrm{U}(1) = \{e^{i\alpha}, -e^{-i\alpha}\}$ for $\alpha = \arcsin(c)$. □

Theorem 3 describes the space of stationary points for infinitely differentiable, surjective, and constant rank mappings. Note that the image of points that do not have constant rank has measure 0 in $\mathrm{Im}(V) = \mathrm{SU}(d)$ by Sard's theorem [38, Ch. 6], so this assumption is not restrictive. Moreover, we have from Theorem 1 that V_{ct} is infinitely differentiable, so we already have a relevant mapping that is infinitely differentiable. Finally, the surjectivity assumption appears as though it could be relaxed. But, unfortunately, this is not the case.

Remark 3. One might wonder if the surjectivity assumption can be relaxed to the assumption $U \in \mathrm{Im}(V)$. The answer is no. We found counter-examples when we were running the numerical experiments. In particular, for certain structures ct with length smaller than the TLB; hence, lacking surjectivity, and with the Toffoli gate as the target unitary U, we computed some stationary points of f_{ct} that exactly compiled U and some that failed the eigenvalue condition of Theorem 3. In other words, for these examples, $U \in \mathrm{Im}(V_{\mathrm{ct}})$, but the conclusion of Theorem 3 does not hold. These results are shown in Table 2: The stationary points in the unsuccessful compilations did not satisfy the eigenvalue condition of Theorem 3.

Since V_{ct} is infinitely differentiable, if only it is surjective as well then its stationary points are precisely the points such that $V_{\mathrm{ct}}(\boldsymbol{\theta})^\dagger U$ has eigenvalues in $\{e^{i\alpha}, -e^{-i\alpha}\}$ for some $\alpha \in [0, 2\pi)$. Having more than one eigenvalue means that the stationary points are not in general global minima up to a global phase transformation though. But, we conjecture that the form of stationary points in Theorem 3 simplifies even further for second-order stationary points of V_{ct}.

Conjecture 1. *If the parametric circuit $V_{\mathrm{ct}}(\cdot)$ is surjective and the Hessian of the cost function $f_{\mathrm{ct}}(\cdot) = \frac{1}{2}\|V_{\mathrm{ct}}(\cdot) - U\|_{\mathrm{F}}^2$ is positive semi-definite at a stationary point $\boldsymbol{\theta}$, then $V_{\mathrm{ct}}(\boldsymbol{\theta})^\dagger U = e^{i\alpha} I$ for some $\alpha \in [0, 2\pi)$, meaning that any stationary point $\boldsymbol{\theta}$ is a global minimum of the cost function $f_{\mathrm{ct}}(\cdot)$, and therefore a solution to the quantum compiling problem, up to a global phase.*

We tested Conjecture 1 extensively. In order to falsify the conjecture, we would need an example of V_{ct}, $\boldsymbol{\theta}$, and U such that V_{ct} is surjective, the Hessian is positive semi-definite, and $V_{\mathrm{ct}}(\boldsymbol{\theta})^\dagger U$ is not a scalar matrix. We searched for counter-examples in two experiments. In both, we used structures with length smaller than the TLB as the non-surjective mappings, and the QSD decomposition structure as the surjective mapping. In the first experiment, we computed $\boldsymbol{\theta}$ via gradient descent. Hence, there was no need to explicitly compute the Hessian. In the second experiment, we randomly generated $W \in \mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \cap \mathrm{SU}(d)$ and $\boldsymbol{\theta} \in \mathbb{R}^p$, then set $U = V_{\mathrm{ct}}(\boldsymbol{\theta})W$. From the proof of Theorem 3, we have that $\mathfrak{su}(d)^{\perp_{\mathbb{R}\mathrm{e}\langle\cdot,\cdot\rangle}} \cap \mathrm{SU}(d)$ is the set of matrices that are unitarily diagonalizable with eigenvalues in $\{e^{-\alpha}, -e^{-i\alpha}\}$ for some α such that the multiplicity of $e^{i\alpha}$ times α plus the multiplicity of $-e^{-i\alpha}$ times $\pi - \alpha$ is an integer multiple of 2π. So, in order to generate W, we sampled $Q \in U(d)$ randomly and chose a non-scalar eigenvalue matrix Λ, then set $W = Q^{\dagger}\Lambda Q$. Thus, $V_{\mathrm{ct}}(\boldsymbol{\theta})^{\dagger}U$ was a non-scalar matrix by construction. We did not find any counter-examples in either experiment. Note that we did have to explicitly compute the Hessian and its eigendecomposition for the second experiment, but this is the only place in the paper where we actually compute the Hessian.

Conjecture 1 says that, despite the difficulty in finding global minima due to non-convexity, the stationary points we find are global minima up to a global phase. However, this is only in the case of surjectivity, and so the non-convexity causes greater difficulty for lengths shorter than the minimum required for surjectivity. We discuss this issue further in the next section. In particular, the divide between easy compilation problems and harder ones is the surjectivity of $V_{\mathrm{ct}}(\boldsymbol{\theta})$.

6 SPECIAL STRUCTURES

While the results and discussion in the previous section were for arbitrary CNOT unit structures, we would like to focus on particular structures henceforth. The properties that we desire such structures to have are

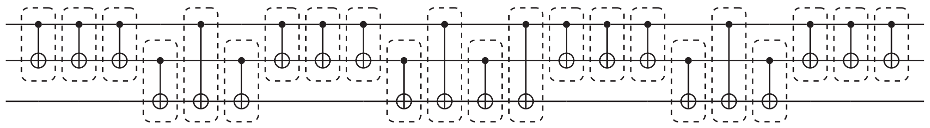
(1) They have to be able to capture all circuits of a given qubit size (i.e., surjectivity).
(2) Their length has to be between one and two times the lower bound for surjectivity.
(3) They have to be compressible, that is, they have to include a method for finding new structures with shorter length.
(4) Their maximum approximation error has to depend favorably on the compressed length.
(5) They should exactly compile special quantum gates (e.g., Toffoli gates) with length close to optimal.
(6) Incorporating hardware constraints should only increase the length of the compressed structure by a constant multiplicative factor in order to preserve the same approximation error.

Many of these properties are intertwined. For example, property (1) is necessary for property (4) and also for convergence (to a stationary point that only requires a global phase transformation), as discussed in the previous section. On the other hand, since the compressed structure will not be surjective when its length is less than the surjectivity bound, convergence becomes more difficult. Assuming property (2), then properties (1) and (5) are the same for all unitary matrices except a measure 0 set. However, there are important matrices in the measure zero set, such as qubit permutations, controlled operations, and cyclic shifts [8], so we keep these requirements separate. But, as mentioned in Remark 3, if we want to exactly compile these with minimal length, the mapping may be far from surjective and so convergence will be much more difficult. We will see this when compiling for Toffoli gates.

We will consider three structures in this paper: cart, standing for Cartan; sequ, standing for sequential; and spin, standing for spin.

Definition 4. Let cart $\in J(\frac{23}{48}4^n - \frac{3}{2}2^n + \frac{4}{3})$ correspond to the QSD decomposition.

As an example, the structure of CNOT units for cart(3) is



.

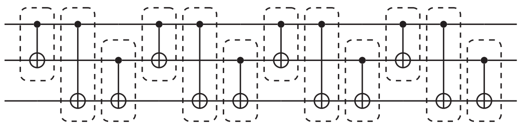
where the CNOT boxes represent our CNOT unit [21]. cart satisfies properties (1) and (2) by the constructive proof of [63]. On the other hand, as is, it does not satisfy property (3) on compressibility. So, we will propose a method in Section 8 that can be used to compress it to as short as the TLB with practically no error, thereby also supporting property (4). cart only increases by a multiplicative factor of 9 for the uncompressed structure, as shown in [63]. However, it is an open problem how to compile special gates close to their optimal length, and compress it in a way that does not increase connectivity, so the answer to properties (5) and (6) is unknown.

While cart has a recursive structure, the next two layouts we consider have repeating structures.

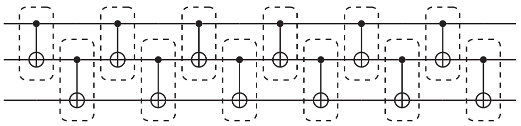
Definition 5. Let $\text{sequ}(L) \in J(L)$ correspond to L CNOT units in order. There are $\frac{n(n-1)}{2}$ CNOT units and the order of $CNOT_{jk}$ is $n(j-1) + \left(k - \frac{j(j+1)}{2}\right)$. Once the end of the order is reached, it repeats.

Definition 6. Let $\text{spin}(L) \in J(L)$ correspond to the structure that alternates between $(1 \rightarrow 2, 3 \rightarrow 4, \ldots)$ and $(2 \rightarrow 3, 4 \rightarrow 5, \ldots)$.

As an example, the structure of CNOTs for $\text{sequ}(12)$ when $n = 3$ is



and for $\text{spin}(12)$ when $n = 3$ is.



Both $\text{sequ}(L)$ and $\text{spin}(L)$ have the length of the circuit as an input, and so satisfy property (3). But for what L, if any, do they satisfy property (1) on surjectivity? We run experiments in Section 7 to evaluate $\text{sequ}(L)$ and $\text{spin}(L)$ with respect to properties (1), (2), and (4). The results support the following conjecture about (1) and (2).

Conjecture 2. *$V_{\text{sequ}(L)}$ and $V_{\text{spin}(L)}$ are surjective for $L \geq \frac{1}{4}(4^n - 3n - 1)$, that is, the TLB.*

In Section 7, we also apply $\text{sequ}(L)$ and $\text{spin}(L)$ to specific quantum gates (e.g., Toffoli), thereby supporting property (5), despite non-surjectivity.

Structure $\text{sequ}(L)$ can incorporate hardware constraints by simply skipping the CNOTs for unconnected qubits, but it is not obvious how much L would have to increase to preserve a given maximum approximation error.

Fortunately, the results in the next section suggest L would not have to increase at all. In the next section, we test $\text{sequ}(L)$ with hardware constraints corresponding to both a star topology and a line topology. All three structures perform comparably to $\text{sequ}(L)$ without any hardware constraints. Furthermore, $\text{spin}(L)$, another structure that satisfies the line topology hardware constraints, also performs comparably. The experiments suggest that different repeating structures, as long as they include all of the qubits, may fill the space equally fast, on average.

In Table 1, we summarize our main findings for the particular structures we have discussed.

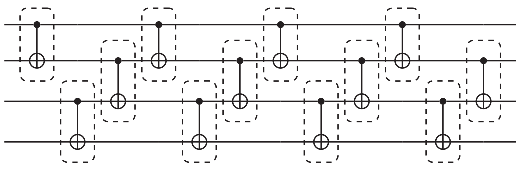
Remark 4. Quantum computers can implement arbitrary gates on disjoint qubits simultaneously. Hence, they can implement CNOT units on disjoint qubits simultaneously. Thus, while our theory

Table 1. Considered Structures and their Properties

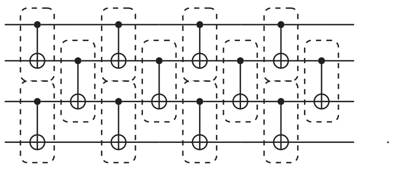
Structure	Properties					
	(1)	(2)	(3)	(4)	(5)	(6)
cart	[21]	[21]	8.3	8.3	?	?
sequ(L)	Conjecture 2		✓	7.2	7.3	7.2
spin(L)	Conjecture 2		✓	7.2	7.3	✓

Numbered references are to subsections.

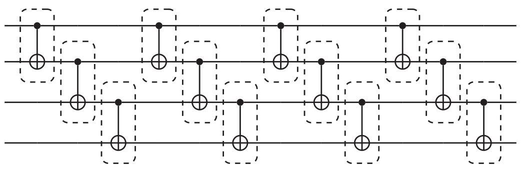
is in terms of length–the number of CNOTs in a structure, two other relevant metrics are CNOT depth—the minimum number of layers of CNOT units in a structure—and circuit depth—the minimum number of layers in a structure. Note that if all rotation angles are non-zero, then circuit depth is three plus three times CNOT depth. Also, CNOT depth is always smaller than or equal to the length. In particular, spin(L) can be implemented with CNOT depth $\lceil 2L/(n-1)\rceil$ for $n > 3$. As another example, spin(12) when $n = 4$ is



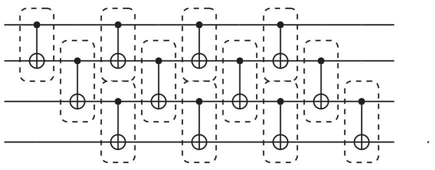
which becomes, if we implement CNOT units on disjoint pairs of qubits simultaneously,



On the other hand, imposing the line topology hardware constraints on sequ(12) results in



which becomes, if we implement CNOT units on disjoint pairs of qubits simultaneously,



Structurally, spin(12) and sequ(12) are almost identical on the line topology, except that spin(12) moves the last CNOT unit of spin(12) to the first layer and so decreases the CNOT depth from 9 to 8.

7 GRADIENT DESCENT

In Section 3, we formulated the mathematical optimization problem, and in Section 5, we discussed some of its properties once specified to CNOT unit structures. In the previous section, we discussed some special layouts. We are now ready to look at the approximate compiling problem for a fixed structure. Specifically, we look at the problem

$$\textbf{(AQCP-}\boldsymbol{\theta}\textbf{)} \qquad \min_{\boldsymbol{\theta}\in[0,2\pi)^p,\,\mathrm{ct}=\overline{\mathrm{ct}}} f_{\mathrm{ct}}(\boldsymbol{\theta}) := \frac{1}{2}\,\|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_{\mathrm{F}}^2, \tag{7}$$

where ct is fixed to one of the mentioned structures (cart, $\mathsf{sequ}(L)$, $\mathsf{spin}(L)$), which, in turn, specifies how many rotation angles there are (i.e., p), and we will use a first-order method to find second-order stationary points. The specific method can be tuned in practice, so we present gradient descent and then discuss the modifications that can be made for its variants.

First, we randomly initialize $\boldsymbol{\theta}[0]$ from the uniform distribution on $[0, 2\pi)^p$. Then, we compute

$$\boldsymbol{\theta}[t+1] = \boldsymbol{\theta}[t] - \alpha\nabla f_{\mathrm{ct}}(\boldsymbol{\theta}[t]), \tag{8}$$

until the stopping criteria, $\|\nabla f_{\mathrm{ct}}(\boldsymbol{\theta}[t])\| \le \epsilon$, is met. The step-size, α, is tuned in practice, and the final $\boldsymbol{\theta}$ is then wrapped in the $[0, 2\pi)^p$ set. Component-wise, Equation (8) reads,

$$\theta_k[t+1] = \theta_k[t] + \alpha \mathbb{R}\mathrm{e}\,\mathrm{Tr}\left[\frac{\partial}{\partial\theta_k} V_{\mathrm{ct}}(\boldsymbol{\theta})^\dagger U\right]. \tag{9}$$

One variant of gradient descent is Nesterov's method [51], which applies the gradient descent step to an auxiliary sequence. The auxiliary sequence is essentially the main sequence but with so-called "momentum." For smooth convex optimization, Nesterov's method is optimal among first-order methods.

Another modification that can be made to gradient descent is the injection of noise. On one hand, diminishing Gaussian noise can be added to the gradient, in which case the method is called gradient descent with Langevin dynamics. On the other hand, noise uniformly sampled from a fixed radius ball can be added in a random direction whenever a "stuck" criteria is met, in which case the method is called perturbed gradient descent.

We found that it was not necessary to add noise and that Nesterov's method was much faster than gradient descent, so we used it as our method of choice. As mentioned previously, random initialization helps escape saddle points to find second-order stationary points. However, the random prior may also affect what kind of second-order stationary points are found. In the non-surjective setting, we would like to avoid spurious local minima as well as saddle points, so it is a future research direction to consider different initializations.

7.1 A Note on Computational Complexity

Note that the gradient computation makes up the majority of the computation and memory complexity. Specifically, we have to compute $\frac{\partial}{\partial\theta_k}V_{\mathrm{ct}}(\boldsymbol{\theta})$ for each $k \in \{1,\ldots,p\}$, each of which takes the same number of flops to compute as $V_{\mathrm{ct}}(\boldsymbol{\theta})$. To compute the matrix for a single layer, via the kronecker product, takes $O(d^2)$ flops. To multiply two matrices takes $O(d^3)$ flops. There are L layers so the total computational complexity for $V_{\mathrm{ct}}(\boldsymbol{\theta})$ is $O(Ld^3)$. Thus, computing $\frac{\partial}{\partial\theta_k}V_{\mathrm{ct}}(\boldsymbol{\theta})$ from scratch for each $k \in \{1,\ldots,p\}$ comes out to $O(L^2d^3)$. The memory complexity is $O(L+d^2)$. Since this is the dominant part of the algorithm, the total computational complexity is $O(L^2d^3T)$ (where T is the number of iterations) and the memory complexity is $O(L+d^2)$.

Fortunately, it is possible to trade-off between the computational and memory complexity. We do this in a similar way to backpropogation, the algorithm for computing the gradient of the loss of a feedforward neural network applied to a sample point [26], which is an example of reverse mode automatic differentiation [40]. Instead of computing $\frac{\partial}{\partial \theta_k} V_{\text{ct}}(\boldsymbol{\theta})$ from scratch for each $k \in \{1, \ldots, p\}$, we can store each of the intermediate matrix computations. Specifically, we can compute the matrix for each layer, taking $O(Ld^2)$ flops and storage. Then, we can compute the matrix multiplications from both the right and left, storing the intermediate outputs. That takes $O(Ld^3)$ flops and $O(Ld^2)$ storage. Then, we compute four new versions of each layer corresponding to the partial derivative of each of the four parameters in a layer. This takes $O(Ld^2)$ flops and storage again. Finally, for the partial derivative of each parameter, we make only two matrix computations, taking $O(Ld^3)$ flops and $O(d^2)$ storage (since we do not store all p matrices at once, we multiply them by U and compute the real part of the trace to get a real number). Thus, this way of computing the gradient descent iterates amounts to $O(Ld^3T)$ flops and $O(Ld^2)$ storage. The reduction in computational complexity from L^2 to L is significant since L can be exponential in the number of qubits. In particular, when the goal is exactly compiling general unitary matrices, we need $L = \Omega(4^n) = \Omega(d^2)$. On the other hand, if the goal is exactly compiling specific unitary matrices or approximately compiling general unitary matrices, L may be smaller. For example, to exactly compile the Toffoli gate, we only need $L = \Omega(n) = \Omega(\log_2(d))$.

These analyses of the computation and memory complexity of gradient descent are for its implementation on a classical computer. However, it can be implemented on a quantum computer as well. Algorithm 3 of [33] explains how to implement it using the power-of-two-qubits circuit. While this is a possibility, we do not follow it here.

7.2 Numerical Tests: Random Unitary Matrices, Toward [G1]

We start by testing the structures and gradient descent on random unitary matrices, which is our goal [G1]. We consider three different hardware connectivity graphs with edge sets

$$E_1 = \{(j,k) \mid j,k \in \{1,\ldots,n\}, j \neq k\}, \qquad \text{(Full connectivity)}$$
$$E_2 = \{(1,j+1) \mid j \in \{1,\ldots,n-1\}\}, \qquad \text{(Star connectivity)}$$
$$E_3 = \{(j,j+1) \mid j \in \{1,\ldots,n-1\}\}. \qquad \text{(Line connectivity).}$$

E_1 corresponds to no hardware constraints, E_2 corresponds to the star topology, and E_3 corresponds to the line topology. Corresponding to these, we consider the structures sequ_{E_1}, sequ_{E_2}, sequ_{E_3}, and spin. Note that spin corresponds to E_3, but $\text{sequ}_{E_3} \neq$ spin except when $n \leq 3$.

Full connectivity results. We ran the experiments for $n = 3$ and $n = 5$ qubits. In both cases, we sampled 100 random unitary matrices for each structure for different values of L. We ran gradient descent for each sample and then computed the approximation error and its probabilities. These curves are given in Figures 4 and 5. As one can see, both sequ and spin are almost identical on how the error depends on the length L, thereby suggesting that the actual structure does not make a big difference. In addition, both structures reach zero error around their theoretical lower bound (which can be computed as $L = 14$ for $n = 3$, and $L = 252$ for $n = 5$), supporting our Conjecture 2. Figure 5 gives probabilistic statements: The x-axis represents a desired maximum approximation error (bound), while the y-axis reports the probability that starting from a random guess for a random unitary, we obtain an error larger than the desired bound. Note that this corresponds to one minus the CDF of the approximation error. Each L considered has a curve in Figure 5, representing the distribution of the approximation error, while in Figure 4, it is compressed to a single point by taking either the mean or the maximum.

Limited connectivity results. With the same settings, we investigate the effect of different connectivity patters on sequ, for $n = 5$ qubits. For $n = 5$, the theoretical lower bound is 252 CNOTs. Figure 6 shows that sequ on all three connectivity patterns approach zero approximation error

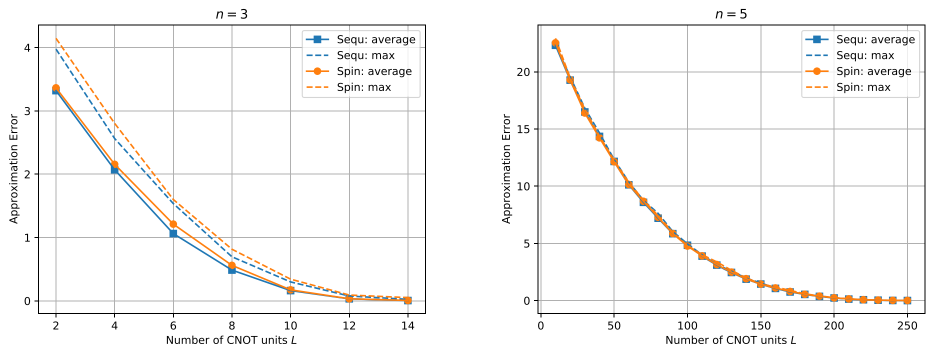


Fig. 4. Approximation error, f_{ct}, for $n = 3$ (left) and $n = 5$ (right).

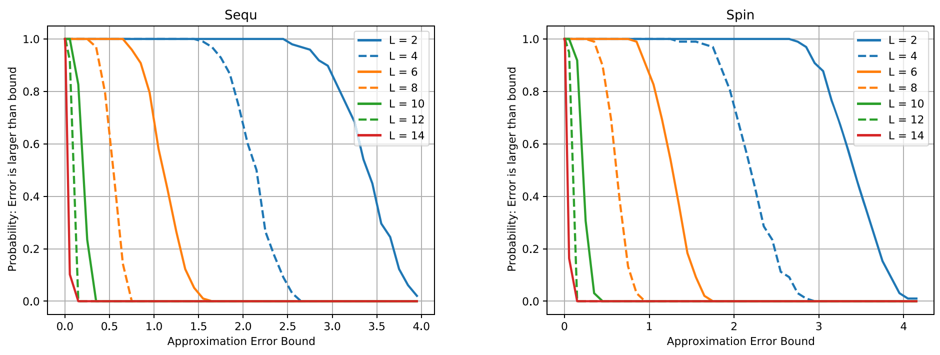


Fig. 5. Error probabilities for sequ and spin on 3-qubits for different values of L that starting from a random initial point for a random unitary we obtain an approximation error larger than a desired bound.

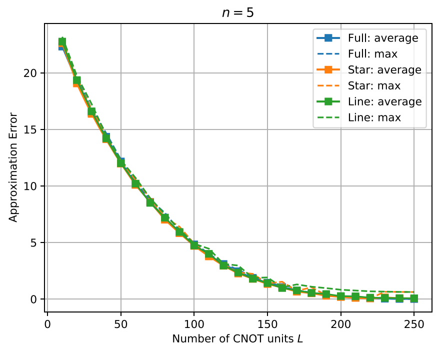
around its theoretical lower bound (in the worst case, the max lines arrive at about 0.7 error, which corresponds to a fidelity of 96%). Furthermore, the dependence of the approximation error on the length is a convex curve that is basically identical for all three patterns.

From an engineering perspective, it is quite surprising that the structures with limited hardware connectivity perform as well as sequ, corresponding to full connectivity. However, from the "separating parameters" perspective that was used to derive the lower bound, this is less surprising: sequ seem to "fill out" the quantum circuit and separate the parameters equally as well on a line as on a fully connected graph.

This is in sharp contrast with the QSD, which uses 20 and 444 CNOT units for $n = 3$ and $n = 5$, respectively, and increases in length by a multiplicative factor of 9 for the line topology.

These experiments suggest that both spin and sequ are surjective for L equal to the lower bound and that their approximation error depends favorably on the length, supporting properties (1) through (4). Regarding property (6), the experiments suggest that the length does not have to increase at all (or perhaps very slightly, in the worst case scenario).

Remark 5. It is important to note here that the fact that the connectivity does not matter in a random unitary setting is a general statement about the overall smooth optimization landscape. We are saying that, if you take a random unitary, and a few random initial points, on average, the approximation error you obtain depends only on the number of CNOT units. This statement says that even if the different structures have different local minima and properties, with a macro-scale



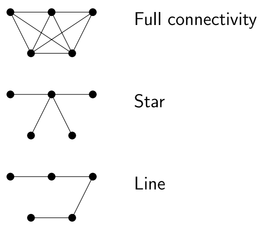


Fig. 6. Approximation error $n = 5$ with different connectivity patterns.

lens (i.e., in the worst-case), they behave very similarly. On the other hand, when one is interested in compiling a particular gate (say a Toffoli), in the least number of CNOT units, the structure plays an important role as discussed in the literature, e.g., [7, 39], and as we will see next. In this case, the fact that connectivity matters is a statement about best case scenarios.

7.3 Numerical Tests: Special Gates, Toward [G2]

The only property we are left to explore in this section is property (5), meaning how close to optimal are the various structures in the case of special (often used and well-studied) gates. So, in addition to compiling random unitary matrices as efficiently as possible, we want to recover the shorter lengths that some important gates allow. For example, k-controlled $(n-k)$-qubit operations can be compiled with $O(n)$ gates if $k < n-1$ and $O(n^2)$ gates if $k = n-1$ [8, Ex. 4.2][4, Lems. 7.2 and 7.5]. In particular, the n-qubit Toffoli gate, also known as the multi-controlled-X gate, can be thought of as a k-controlled $(n-k)$-Toffoli gate, and so can be compiled with $O(n)$ gates. Furthermore, [64] gives a lower bound of $2n$ CNOTs for the n-qubit Toffoli gate (other lower bounds are possible when considering circuits with ancillae [41]). The lower bound is tight for $n = 3$. Qiskit [56] decomposes the 3-qubit Toffoli gate into 6 CNOTs and the 4-qubit Toffoli gate into 14 CNOTs.[2] We remark that these numbers of CNOTs are significantly lower than the TLB, and therefore property (5) is not trivial to fulfill.

In addition to the Toffoli gate, we also consider the Fredkin 3 qubit gate, as well as the 1-bit full adder (which is a 4 qubit gate). The list of important gates is by no means exhaustive, but it already gives a glimpse of how well sequ and spin work for important gates.

To test property (5), we ran our gradient descent algorithm on spin and sequ many times (we report in Table 2 the results, as well as the number of exact compilations divided by the number of tries). We compare with the Qiskit compilations on full connectivity and on line connectivity (with the usual workflow, one would compile, e.g., a Toffoli gate on the full connectivity hardware and then add swap gates to transpile it on the limited connectivity one) [56].

In the case of sequ, we also consider the case of permuting the CNOT units to span more possibilities: This is, at the moment, a random search permuting all the possibilities, which is unpractical for large L's (for instance for $L = 14$ it would amount to over 150 million possibilities), but gives us a glimpse that structure does matter when compiling very specific gates in the non-surjective domain.

[2]A 14 CNOT implementation can be obtained from the 20 CNOT one of [4] by substituting the controlled-Vs with their 2-CNOT implementations and applying the templates presented in [42].

Table 2. Exact Compilations of Special Gates

Gate	Qiskit CNOTs		sequ		spin
	Full conn.	Line conn.	w/o perm.	w perm	
Toffoli 3 qubit	6	7–9	7 (19/100)	6 (27/100)	8 (38/100)
Toffoli 4 qubit	14	36	18 (4/350)	14 (1/100)*	18 (1/350)
Fredkin 3 qubit	7	10	8 (55/100)	7 (4/100)	8 (31/100)
1-bit full adder 4 qubit	10	16	10 (11/100)	10 (3/100)	14 (8/500)

In parentheses, the number of successful compilations vs. the number of trials starting with a different initial condition, and in the case of sequ with permutation, with a different permutation of the CNOT layout. Note that the Qiskit compilation (opt level of 3 and "sabre" as routing and layout) is stochastic and can return a variable number of CNOTs. *After several trials, we have found a satisfactory layout, and the numbers correspond to this one.

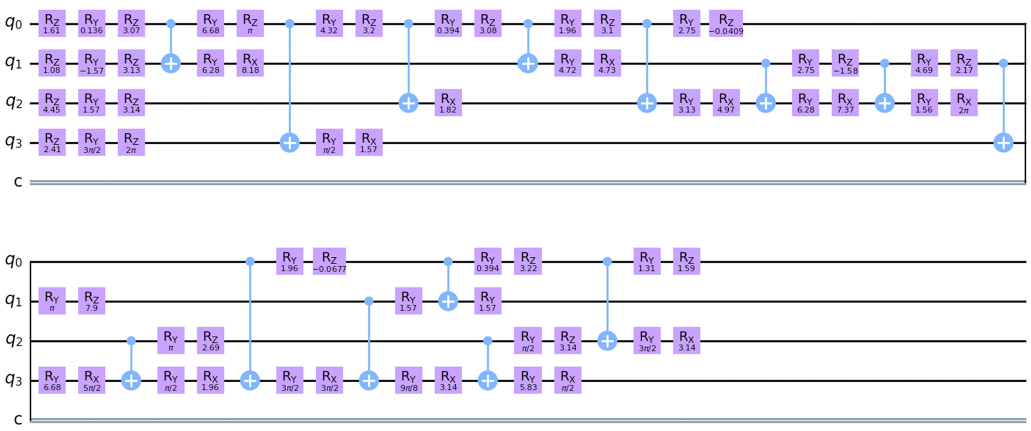


Fig. 7. A different from Qiskit 4-qubit Toffoli exact compilation, obtained by permuting the CNOT units in the sequ(14) structure and running gradient descent with many initial points.

The results indicate that, e.g., sequ(18) and spin(18) can exactly compile a 4-qubit Toffoli, which is better than the 20 CNOT implementation of [4]. And allowing for permutations in the sequence of CNOTs, even sequ(14) can do it, which is optimal.[3] The solution we have found is reported in Figure 7, and it uses a different sequence of CNOT gates than the one in Qiskit.

What we observe is that there are different possibilities in compiling a certain circuit exactly, and in most cases the optimal solutions are different. This shows that our algorithm can be used as a tool to discover new exact compilations of special gates.

The experiment suggests that property (5) is satisfied to a reasonable extent for both sequ and spin.

Finally, by looking at the performance of Qiskit when compiling on a line connectivity and comparing it to our spin structure, which enforces the line connectivity by design, we further appreciate the advantage of our method in terms of CNOT count.

[3]The ease at which we have found this, despite the 14!/3!/3!/2!/2!/2!/2! possible structures, indicates that there may be more than one structure that can deliver an optimal compilation. Note that, in general, for m types of CNOTs with corresponding quantities $L_1, \ldots, L_m$, there are $(L_1 + \cdots + L_m)!/(L_1! \cdots L_m!)$ permutations.

8 CIRCUIT COMPRESSION VIA REGULARIZATION

We move now to address property (3) on compressibility in more detail. In particular, motivated by the cart structure, we notice that, while cart is provably surjective with length twice the lower bound, thus satisfying properties (1) and (2), it is not clear how to compress it and so address property (3). The purpose of this section is to develop an algorithmic technique for compressing arbitrary structures, and cart in particular. Rather than choosing which CNOTs to keep in a structure (equivalently, which CNOTs to eliminate) *a priori*, we consider the question of how to design an algorithm that automatically finds the best compression for the target unitary U.

We approach this problems with two ideas: (1) We know that there are techniques to reduce consecutive CNOTs when no rotation gates are between them (let us call these compaction rules); (2) We know that, when optimizing for the angles, we can enforce sparsity of the solution by adding a pertinent regularization.

A closer look at (2) inspires us to enforce groups of four rotation angles following the CNOTs to be zero, thereby eliminating all the rotation gates after a CNOT. This leads naturally to a group LASSO regularization, and we explore it in Section 8.2.

A closer look at (1) suggests special compaction rules, which we discuss in Section 8.1.

Finally, the complete algorithm starts by enforcing sparsity, eliminating zero rotation gates, compressing the structure via compaction rules, and then re-optimizing with the regular gradient descent on the compacted structure. This algorithm is discussed in Section 8.3.

8.1 The "Synthesis" Algorithm

The guiding question for this subsection is: given a list of CNOTs, can we find a shorter list of CNOTs such that the matrix product, in $\mathrm{SU}(2^n)$, of the first list and the second list are equal. While the research works in this area are many [1, 2, 8, 12, 23, 44, 54], we use here an adapted version of the "synthesis" algorithm of [54]. There the authors give an asymptotically optimal synthesis of CNOT circuits. The idea is that CNOTs on n-qubits can be identified with elementary matrices in $\mathrm{GL}(n, \mathbb{Z}_2)$. Given $A \in \mathrm{GL}(n, \mathbb{Z}_2)$, we can compute its LU decomposition in terms of elementary matrices. On the other hand, things are easier in our case, since we only consider downward-facing CNOTs, which can be related to $\mathrm{LT}(n, \mathbb{Z}_2)$, the group of n by n lower unitriangular matrices on $\mathbb{Z}_2$. This leads us to implement an adapted "synthesis" algorithm that consists of three steps: identifying CNOTs with their corresponding matrices in $\mathrm{LT}(n, \mathbb{Z}_2)$, multiplying them in $\mathrm{LT}(n, \mathbb{Z}_2)$, and reading off the locations of 1's in the product. We report the following theorem concerning this algorithm.

Theorem 7. *[54] Given an L-long circuit of downward-facing CNOTs, the "synthesis" algorithm correctly outputs an equivalent circuit of downward-facing CNOTs of length $\leq n(n-1)/2$ in $O(n^3 L)$ computations.*

The correctness of the algorithm follows from [54], the length follows from the fact that there are $n(n-1)/2$ lower-triangular entries in an n by n matrix, and the run-time is based on $L-1$ matrix multiplications. While the algorithm may output a word of length $n(n-1)/2$, it is possible for words to be further reduced. Hence, this "synthesis" algorithm is not optimal, but its simplicity is appealing. The lower bound on word lengths is $\Omega(n^2/\log(n))$ for $\mathrm{GL}(n, \mathbb{Z}_n)$, and one way to obtain it is to partition the matrix into blocks, as is done in [54]. Note that words can also be reduced via identities on three qubits, namely, commutation rules, cancellations of two subsequent identical CNOTs, and mirror rules (see, e.g, [22]). However, these have little effect for $n > 3$.

We remark that, in general, the "synthesis" algorithm does not respect hardware connectivity, so it will be used only in cases in which hardware connectivity is not an issue. However, the three qubit identities do maintain the hardware connectivity constraints of the original circuit, so they can be used in all the cases. Note that there are recent works using Steiner trees to apply the synthesis algorithm in a way that does respect hardware connectivity: [20, 25, 34, 50, 71].

8.2 Setting Parameters to Zero

The guiding question for this subsection is: Which parameters should we set to 0 to get a good compressed structure via the techniques presented in the previous subsection? As mentioned, our approach is to enforce sparsity of the resulting $\boldsymbol{\theta}$ vector by pushing groups of the four-rotation angles following a CNOT to be 0 (i.e., all the rotations of a given CNOT unit). This can be achieved via a group Lasso regularization [66] (see also [5, 6, 14]), as follows. The collection of rotation angles for each of the CNOT units is the vector $\boldsymbol{\theta}_\ell := [\theta_{3n+4\ell-3}, \dots, \theta_{3n+4\ell}]$, with $\ell = \{1, \dots, L\}$. Enforcing each of these vectors to be 0, amounts to adding a regularization of the form $\|\boldsymbol{\theta}_\ell\|_2$ for each group, and therefore solving the problem:

$$\textbf{(AQCP-}\boldsymbol{\theta}, \lambda\textbf{)} \qquad \min_{\boldsymbol{\theta} \in [0, 2\pi)^p,\, \mathrm{ct} = \overline{\mathrm{ct}}} f_{\mathrm{ct}}(\boldsymbol{\theta}; \lambda) := \frac{1}{2} \|V_{\mathrm{ct}}(\boldsymbol{\theta}) - U\|_{\mathrm{F}}^2 + \lambda \sum_{\ell=1}^{L} \|\boldsymbol{\theta}_\ell\|_2 \,, \tag{10}$$

with regularization parameter $\lambda > 0$, which trades-off approximation error and sparsity.

We can solve Equation (10) by a proximal gradient descent, as done in [66]; although problem (10) is non-convex, we have similar convergence results as for gradient descent, meaning that for small regularization parameters λ, we can show convergence of perturbed proximal gradient descent to a second-order stationary point [28].

In particular, proximal gradient descent amounts to computing a gradient descent step and then applying a proximal operator (which, in this case, is the block-wise soft-thresholding operator [45, Equation (7)]). Starting from a randomly initialized $\boldsymbol{\theta}[0]$, this yields the component-wise recursion for $k = 3n + 4\ell - 3, \dots, 3n + 4\ell$, for $\ell = 1, \dots, L$, as

$$[\boldsymbol{\theta}_\ell^+]_k = \theta_k[t] + \alpha \mathrm{Re}\, \mathrm{Tr}\left[\frac{\partial}{\partial \theta_k} V_{\mathrm{ct}}(\boldsymbol{\theta}[t])^\dagger U\right] \quad \text{for all } k \in \{3n + 4\ell - 3, \dots, 3n + 4\ell\}, \tag{11a}$$

$$\theta_k[t+1] = \frac{[\boldsymbol{\theta}_\ell^+]_k}{\|\boldsymbol{\theta}_\ell^+\|} (\|\boldsymbol{\theta}_\ell^+\| - \alpha\lambda)_+ \qquad \text{for all } k \in \{3n + 4\ell - 3, \dots, 3n + 4\ell\}, \tag{11b}$$

$$\theta_k[t+1] = \theta_k[t] + \alpha \mathrm{Re}\, \mathrm{Tr}\left[\frac{\partial}{\partial \theta_k} V_{\mathrm{ct}}(\boldsymbol{\theta}[t])^\dagger U\right] \qquad \text{for all } k \in \{1, \dots, 3n\}, \tag{11c}$$

where $(y)_+ = \max\{y, 0\}$.

Equation (11) is block-wise recursion: For each CNOT unit, we run the gradient descent for each angle of the unit, and then run the proximal operator. For the angles not belonging to the CNOT units, then it is business as usual.

8.3 Compression Algorithm

Integrating the ideas from the previous two subsections, we propose the following algorithm for compressing an arbitrary structure.

The algorithm consists of running proximal gradient descent (11), starting with some given initial condition $\boldsymbol{\theta}[0]$, on the regularized cost $f_{\mathrm{ct}}(\cdot, \lambda)$. Then, we eliminate all the rotation gates that have 0 rotation angle and apply the compaction rules to reduce the circuit. Finally, we run standard gradient descent (8) on $f_{\mathrm{ct}'}$, where ct′ is the compressed structure, to compute the best parameters $\boldsymbol{\theta}^*$ for the compressed structure.

Line 3 (i.e., "synthesis") can be included, when hardware connectivity constraints are not important, or not, when they are.

ALGORITHM 1: Compression via group LASSO

```
Require: $U$, ct, $\lambda$, initial condition $\boldsymbol{\theta}[0]$
1: compute $\boldsymbol{\theta}^*_{\mathrm{GL}}$ via proximal gradient descent (11) on $f_{\mathrm{ct}}(\cdot, \lambda)$
2: Eliminate all the rotation gates that have zero rotation angle
3: compress $ct$ via the "synthesis" algorithm
4: further compress $ct$ via CNOT identities
5: compute $\boldsymbol{\theta}^*$ via standard gradient descent (8) on $f_{\mathrm{ct}'}$, where ct$'$ is the compressed structure
6: return compressed structure, ct$'$, and corresponding angles, $\boldsymbol{\theta}^*$
```

8.4 Numerical Tests: Random Unitary Matrices, Toward [G3]

We test our compression algorithm on random unitary matrices with sequ and cart structures for both $n = 3$ and $n = 5$ qubit circuits. (We do not consider spin because it corresponds to the line connectivity, which is not preserved under the synthesis algorithm.) In particular, we randomly initialize the iterates and consider 100 different random unitary matrices, and we plot both mean and standard deviation of the result.

We report our results in Figure 8. We start with circuits of $L = 14$ and $L = 250$ for sequ and from $L = 22$ and $L = 528$ for cart, respectively, for $n = 3$ and $n = 5$, and we compress them with different regularization parameters. In the x-axis, we can see the regularization parameter λ used, while in the y-axis, we can note the final number of CNOTs (in blue), as well as the Frobenius fidelity $\bar{F}_{\mathrm{F}}(U, V)$ (in orange). The lines correspond to the average, and the shaded areas to one standard deviation.

As one can appreciate, a small compression is possible for sequ, especially if the "synthesis" algorithm is used. Note that the three qubit reductions have little affect for $n = 5$ and this is true for all $n \geq 5$.

A better compression, with very high Frobenius fidelity $\bar{F}_{\mathrm{F}}(U, V)$, is possible instead for cart, showing that this structure can be tuned to be compressed to TLB ($L = 14$ and $L = 252$, respectively) without losing fidelity. One can see this by looking at the second data point on the left for both $n = 3$ and $n = 5$, where we obtain a reduction to $L = 14$ and $L < 252$ (blue curve), with no practical loss of fidelity (orange curve). This is encouraging (meaning that the recursive Cartan decomposition can be easily compressed in practice) and supports properties (3) and (4) for cart.

8.5 Numerical Tests: Compressing Compiled Circuits, Toward [G3] in Qiskit

Finally, we move to analyze the effect of the compression algorithm to already compiled circuits in Qiskit, or other compilers. The idea here is to see how one can use the approximate quantum compiler as an add-on to the usual workflow, by allowing the user to trade-off accuracy and circuit depth, as defined in Remark 4.

We consider the quantum circuits of 3, 4, and 5 qubits in the [75] database. We compile them in Qiskit (opt level of 3 and "sabre" as routing and layout) and transform them into the parametric circuit of the present article (see Figure 2 for an example). Then, we use this compiled circuit as a warm start for our compression algorithm, Algorithm 1 for different λ's. This yields the graphs in Figure 9, where we can appreciate how λ affects compression, as well as Frobenius fidelity.

A better overview is offered by Table 3, where we look at the best compression we can obtain for a Frobenius fidelity as high as 90% and 80%, and its associated computational overhead. To compile the table, we have looked at different λ's parameters, and we have considered the best compression with fidelity ≥90%, 80%. The column "Depth" refers to the depth of the original circuit

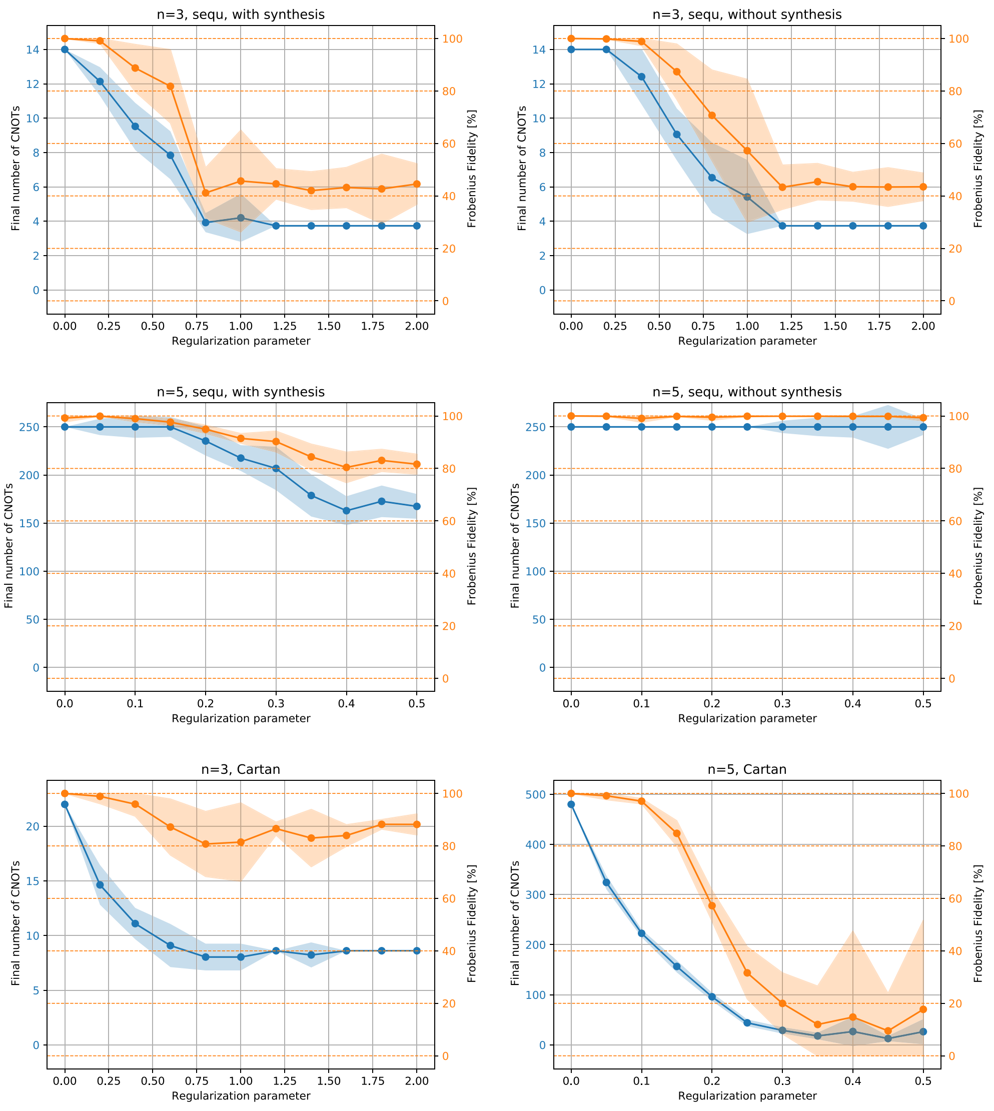


Fig. 8. Final number of CNOT units and Frobenius fidelity, for $n = 3$ and $n = 5$ qubit circuits and different structures. We divide the cases for sequ for when "synthesis" is run, and when it is not. For cart, since hardware constraints are not imposed, "synthesis" is always run. For all graphs, the x-axis represents the regularization parameter λ, while the y-axis represents the compression obtained, in blue, and the Frobenius fidelity, in orange. The continuous line is the average, while the shaded area is one standard deviation.

in the database (which can have gates not in the gate set), the Qiskit depth is the circuit depth once compiled in Qiskit and transformed into the parametric circuit, the compressed depth is the minimal depth that we were able to obtain with Algorithm 1, with a fidelity $\geq 90\%$ (or $\geq 80\%$ – if it is the same depth, we do not report it twice) (by varying λ), the fidelity column is the Frobenius fidelity of the maximal compressed circuit, while the overhead is the time that the computation of the latter circuit has taken.

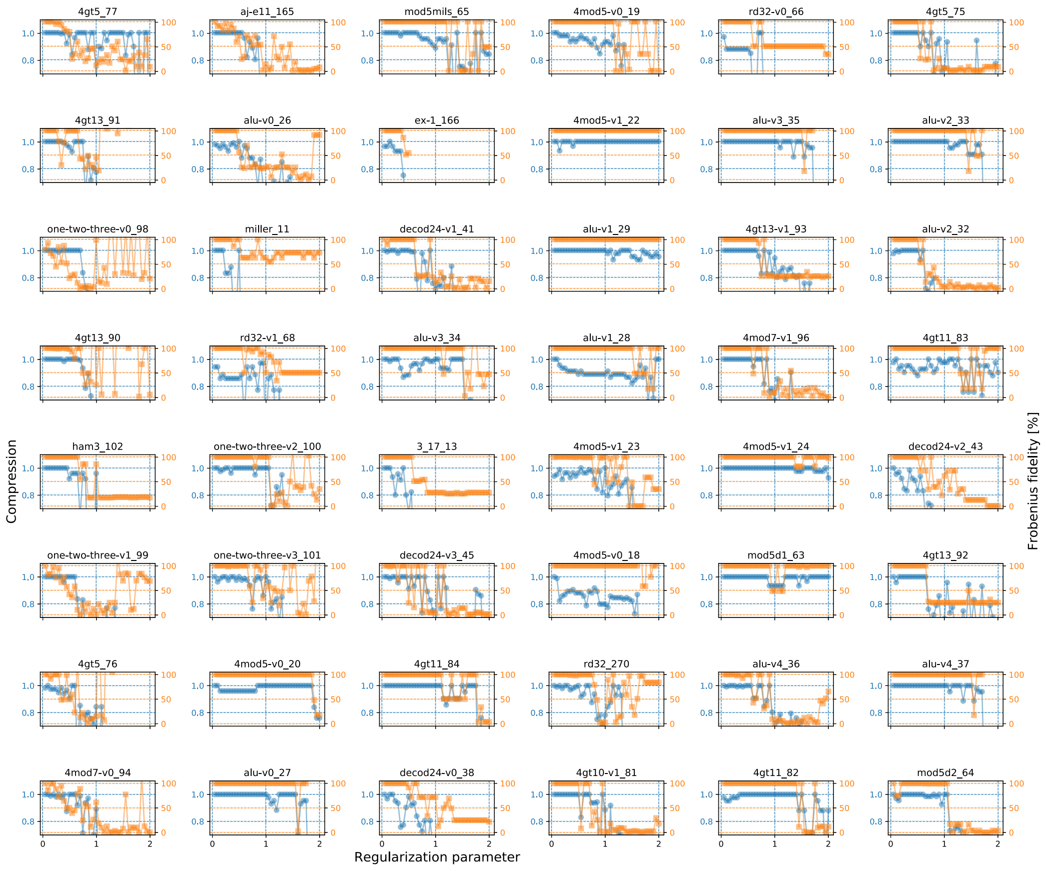


Fig. 9. Compression of compiled circuits from the [75] database, for various regularization parameter values. For all graphs, the x-axis represents the regularization parameter [0–2], while the y-axis the compression obtained in blue (normalized to the starting CNOT count), and the Frobenius fidelity in orange.

As one can see, we can achieve some compression with a small loss in fidelity, and in some cases, without any loss. For some circuits, compression is quite high (∼58% of the Qiskit depth), for others, less so.

We remark that, ultimately, the compression capabilities are a by-product of Qiskit (or others) compilation properties. If the compiler used can achieve optimal compilation, no further compression can be obtained, no matter how sophisticated the devised algorithm is. While compilation is getting better and better (new exact or heuristic algorithms are proposed at a rapid pace, see for instance [49]), we believe that our tool can *also* be used to globally gauge if circuits can be further compressed and which part of the circuit is not optimally compiled. This has immediate practical applications in devising better compilation algorithms. For example, further studies are needed to understand what makes "miller_11" or "decod24-v2_43" difficult for Qiskit, and how to improve its compiler.

Finally, more research has to be dedicated toward devising better strategies to select the best parameter λ for compression, instead of a random search, and analyzing what happens with the use of different compilers rather than Qiskit.

Table 3. Best Achieved Compression for the Circuit of the [75] Database, Along with their Frobenius Fidelity and Computational Overhead

		Qiskit	Fr. Fidelity ≥ 90%		Fr. Fidelity ≥ 80%		
File name	Depth	depth	Compr. depth (& %)	$\bar{F}_F$ [%]	Compr. depth (& %)	$\bar{F}_F$ [%]	Overhead [s]
4gt5_77	74	137	136 (99%)	93.41	126 (92%)	80.6	24.34
4gt13_91	61	119	119 (100%)	100.0			20.76
one-two-three-v0_98	82	163	163 (100%)	100.0			23.7
4gt13_90	65	121	120 (99%)	99.99			20.63
ham3_102	13	25	24 (96%)	99.99			1.28
one-two-three-v1_99	76	152	152 (100%)	100.0			23.65
4gt5_76	56	115	111 (97%)	98.75			20.57
4mod7-v0_94	92	175	171 (98%)	95.22			27.19
aj-e11_165	86	177	177 (100%)	100.0			36.16
alu-v0_26	49	99	95 (96%)	99.99			17.26
miller_11	29	65	38 (58%)	99.99			3.1
rd32-v1_68	21	35	34 (97%)	92.02	31 (89%)	82.43	4.58
one-two-three-v2_100	40	79	79 (100%)	100.0			13.82
one-two-three-v3_101	40	80	79 (99%)	91.84			16.85
4mod5-v0_20	12	25	25 (100%)	100.0			6.01
alu-v0_27	21	43	41 (95%)	99.99			7.8
mod5mils_65	21	44	44 (100%)	100.0			8.61
ex-1_166	12	28	26 (93%)	99.99	21 (75%)	85.34	1.28
decod24-v1_41	50	94	93 (99%)	99.99			16.32
alu-v3_34	30	60	60 (100%)	100.0			11.45
3_17_13	22	45	37 (82%)	99.98			2.11
decod24-v3_45	84	157	155 (98%)	98.4			25.86
4gt11_84	11	21	21 (100%)	100.0			2.82
decod24-v0_38	30	62	52 (84%)	99.99			5.48
4mod5-v0_19	21	45	41 (91%)	95.94			8.76
4mod5-v1_22	12	29	29 (100%)	100.0			6.06
alu-v1_29	22	43	41 (95%)	99.99			6.98
alu-v1_28	22	45	39 (87%)	99.99			7.18
4mod5-v1_23	41	83	76 (92%)	99.99			14.79
4mod5-v0_18	40	84	73 (87%)	99.99			11.06
rd32_270	47	96	84 (88%)	99.81	81 (84%)	89.44	20.94
4gt10-v1_81	84	159	159 (100%)	100.0			29.35
rd32-v0_66	20	33	33 (100%)	100.0			3.73
alu-v3_35	22	44	42 (95%)	99.99			7.77
4gt13-v1_93	39	70	70 (100%)	100.0			14.66
4mod7-v1_96	94	164	164 (100%)	100.0			24.4
4mod5-v1_24	21	41	38 (93%)	97.4			8.68
mod5d1_63	13	30	30 (100%)	100.0			8.92
alu-v4_36	66	117	117 (100%)	100.0			20.24
4gt11_82	20	42	42 (100%)	100.0			7.18
4gt5_75	47	88	88 (100%)	100.0			16.24
alu-v2_33	22	42	38 (90%)	99.99			6.96
alu-v2_32	92	174	174 (100%)	100.0			25.21
4gt11_83	16	41	37 (90%)	99.95			9.18
decod24-v2_43	30	65	47 (72%)	99.99			5.23
4gt13_92	38	71	71 (100%)	100.0			14.1
alu-v4_37	22	44	42 (95%)	99.99			7.73
mod5d2_64	32	67	67 (100%)	100.0			14.08

Depth is the original depth; Qiskit depth is the depth once compiled in Qiskit and transformed into the parametric circuit; Compr. depth is the best compression obtained with Fr. Fidelity ≥90% or ≥80% (in the ≥80% column, we only report depth if different from the one in ≥90%), while its indicated % represents the ratio of the compressed depth w.r.t. the Qiskit depth.

9 CONCLUSIONS AND OPEN POINTS

In this article, we have examined, in deep mathematical and numerical detail, variants of the best approximate quantum compiling problem. We have shown how to build hardware-aware structures and how to optimize over them. While we have presented encouraging results to support various theoretical and numerical properties, several open points are left for future research. In particular, on top of our priority list are (1) to investigate Conjecture 2 and analytically determine the relationship between approximation error and number of CNOTs (which has been done for 2-qubits using the Weyl chamber [17, 18, 32, 55]); (2) to investigate Conjecture 1; and (3) to investigate properties (5) and (6) for cart.

REFERENCES

[1] Scott Aaronson and Daniel Gottesman. 2004. Improved simulation of stabilizer circuits. *Physical Review A* 70, 5 (2004), 052328.

[2] Matthew Amy, Jianxin Chen, and Neil J. Ross. 2017. A finite presentation of CNOT-dihedral operators. In *Proceedings of the International Conference on Quantum Physics and Logic.* 84–97.

[3] M. Amy, D. Maslov, M. Mosca, and M. Roetteler. 2013. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 6 (2013), 818–830.

[4] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. 1995. Elementary gates for quantum computation. *Physical Review A* 52, 5 (1995), 3457.

[5] Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2, 1 (2009), 183–202.

[6] Stephen Becker, Jérôme Bobin, and Emmanuel J. Candès. 2011. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences* 4, 1 (2011), 1–39.

[7] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. 2019. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information* 5, 1 (2019), 45.

[8] Thomas Beth and Martin Rötteler. 2001. Quantum algorithms: Applicable algebra and quantum physics. In *Proceedings of the Quantum Information.* Springer, 96–150.

[9] Debjyoti Bhattacharjee and Anupam Chattopadhyay. 2017. Depth-optimal quantum circuit placement for arbitrary topologies. arXiv preprint arXiv:1703.08540 (2017).

[10] D. Bhattacharjee, A. A. Saki, M. Alam, A. Chattopadhyay, and S. Ghosh. 2019. MUQUT: Multi-constraint quantum circuit mapping on NISQ computers: Invited paper. In *Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design.* 1–7.

[11] Kyle E. C. Booth, Minh Do, J. Christopher Beck, Eleanor Rieffel, Davide Venturelli, and Jeremy Frank. 2018. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. In *Proceedings of the International Conference on Automated Planning and Scheduling.* 366–374.

[12] Sergey Bravyi and Dmitri Maslov. 2021. Hadamard-free circuits expose the structure of the Clifford group. *IEEE Transactions on Information Theory* 67, 7 (2021), 4546–4563.

[13] Stephen S. Bullock and Igor L. Markov. 2003. An arbitrary two-qubit computation in 23 elementary gates or less. In *Proceedings of the 40th Annual Design Automation Conference.* 324–329.

[14] Emmanuel J. Candes, Michael B. Wakin, and Stephen P. Boyd. 2008. Enhancing sparsity by reweighted ℓ-1 minimization. *Journal of Fourier Analysis and Applications* 14, 5–6 (2008), 877–905.

[15] Lukasz Cincio, Yiğit Subaşı, Andrew T. Sornborger, and Patrick J. Coles. 2018. Learning the quantum algorithm for state overlap. *New Journal of Physics* 20, 11 (2018), 113022.

[16] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. 2019. On the qubit routing problem. In *Proceedings of the 14th Conference on the Theory of Quantum Computation, Communication and Cryptography*, Vol. 135. 5:1–5:32.

[17] Gavin E. Crooks. 2020. Gates, states, and circuits. (2020).

[18] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. 2019. Validating quantum computers using randomized model circuits. *Physical Review A* 100, 3 (2019), 032328.

[19] Christopher M. Dawson and Michael A. Nielsen. 2006. The Solovay-Kitaev algorithm. *Quantum Information & Computation* 6, 1 (2006), 81–95.

[20] Arianne Meijer-van de Griend and Ross Duncan. 2020. Architecture-aware synthesis of phase polynomials for NISQ devices. arXiv:2004.06052. Retrieved 11 November 2021 from https://arxiv.org/abs/2004.06052.

[21] Byron Drury and Peter Love. 2008. Constructive quantum Shannon decomposition from cartan involutions. *Journal of Physics A: Mathematical and Theoretical* 41, 39 (2008), 395305.

[22] J. C. Garcia-Escartin and P. Chamorro-Posada. 2011. Equivalent quantum circuits. arXiv:1110.2998. Retrieved 11 November 2021 from https://arxiv.org/abs/1110.2998.

[23] Shelly Garion and Andrew W. Cross. 2020. On the structure of the CNOT-dihedral group. arXiv:2006.12042. Retrieved 11 November 2021 from https://arxiv.org/abs/2006.12042.

[24] Rong Ge, Chi Jin, and Yi Zheng. 2017. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* . 1233–1242.

[25] Vlad Gheorghiu, Sarah Meng Li, Michele Mosca, and Priyanka Mukhopadhyay. 2020. Reducing the CNOT count for Clifford + T circuits on NISQ architectures. arXiv:2011.12191. Retrieved 11 November 2021 from https://arxiv.org/abs/2011.12191.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

[27] Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis* (2nd ed.), Cambridge University Press, Cambridge ; New York.

[28] Zhishen Huang and Stephen Becker. 2019. Perturbed proximal descent to escape saddle points for non-convex and non-smooth objective functions. In *INNS Big Data and Deep Learning Conference*, Vol. 1. Springer, 58–77.

[29] Toshinari Itoko, Rudy Raymond, Takashi Imamichi, and Atsushi Matsuo. 2020. Optimization of quantum circuit mapping using gate transformation and commutation. *Integration* 70 (2020), 43–50.

[30] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. 2017. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, 1724?1732.

[31] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. 2018. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Proceedings of the Conference On Learning Theory*. 1042–1085.

[32] Petar Jurcevic, Ali Javadi-Abhari, Lev S Bishop, Isaac Lauer, Daniela F Bogorin, Markus Brink, Lauren Capelluto, Oktay Günlük, Toshinaro Itoko, Naoki Kanazawa, et al. 2021. Demonstration of quantum volume 64 on a superconducting quantum computing system. *Quantum Science and Technology* 6, 2 (2021), 025020.

[33] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. 2019. Quantum-assisted quantum compiling. *Quantum* 3, 1 (2019), 140.

[34] A Kissinger and A Meijer-van de Griend. 2020. CNOT circuit extraction for topologically-constrained quantum memories. *Quantum Information and Computation* 20, 7&8 (2020), 581–596.

[35] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. 2013. Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates. *Quantum Information & Computation* 13, 7–8 (2013), 607–630.

[36] Anthony W. Knapp. 2013. *Lie Groups Beyond an Introduction*. Vol. 140. Springer Science & Business Media.

[37] Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. 2016. Gradient descent only converges to minimizers *(Proceedings of Machine Learning Research, Vol. 49)*, Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir (Eds.), PMLR, Columbia University, New York, New York, 1246–1257.

[38] John M Lee. 2013. *Introduction to Smooth Manifolds*. Springer.

[39] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. 2017. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3305–3310.

[40] Seppo Linnainmaa. 1976. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics* 16, 2 (1976), 146–160.

[41] Dmitri Maslov. 2016. Advantages of using relative-phase toffoli gates with an application to multiple control toffoli optimization. *Physical Review A* 93, 2 (Feb 2016), 022311.

[42] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne. 2008. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 3 (2008), 436–444.

[43] D. Maslov, S. M. Falconer, and M. Mosca. 2008. Quantum circuit placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 4 (2008), 752–763.

[44] Dmitri Maslov and Martin Roetteler. 2018. Shorter stabilizer circuits via bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory* 64, 7 (2018), 4729–4738.

[45] Sofia Mosci, Lorenzo Rosasco, Matteo Santoro, Alessandro Verri, and Silvia Villa. 2010. Solving structured sparsity regularization with proximal methods. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 418–433.

[46] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. 2019. Full-stack, real-system quantum computer studies: Architectural comparisons and design insights. In *Proceedings of the 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture*. IEEE, 527–540.

[47] Yumi Nakajima, Yasuhito Kawano, and Hiroshi Sekigawa. 2005. A new algorithm for producing quantum circuits using KAK decompositions. *Quantum Information & Computation* 6, 1 (2006), 67–80.

[48] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. 2018. Automated optimization of large quantum circuits with continuous parameters. *npj Quantum Information* 4, 1 (2018), 23.

[49] Giacomo Nannicini, Lev S Bishop, Oktay Gunluk, and Petar Jurcevic. 2021. Optimal qubit assignment and routing via integer programming. arXiv:2106.06446. Retrieved 11 November 2021 from https://arxiv.org/abs/2106.06446.

[50] Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. 2020. Quantum circuit optimizations for NISQ architectures. *Quantum Science and Technology* 5, 2 (2020), 025010.

[51] Y.E. Nesterov. 1983. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady* 269, 3 (1983), 543–547.

[52] Michael A. Nielsen and Isaac Chuang. 2002. *Quantum Computation and Quantum Information.* American Association of Physics Teachers.

[53] Angelo Oddi and Riccardo Rasconi. 2018. Greedy randomized search for scalable compilation of quantum circuits. In *Proceedings of the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer International Publishing, Cham, 446–461.

[54] Ketan N. Patel, Igor L. Markov, and John P. Hayes. 2008. Optimal synthesis of linear reversible circuits. *Quantum Information & Computation* 8, 3 (2008), 282–294.

[55] Eric C. Peterson, Gavin E. Crooks, and Robert S. Smith. 2020. Two-qubit circuit depth and the monodromy polytope. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer International Publishing, Cham, 446–461.

[56] IBM Quantum. 2019. *Qiskit: An Open-source Framework for Quantum Computing*.

[57] Péter Rakyta and Zoltán Zimborás. 2021. Approaching the theoretical limit in quantum gate decomposition. arXiv:2109.06770. Retrieved 11 November 2021 from https://arxiv.org/abs/2109.06770.

[58] Péter Rakyta and Zoltán Zimborás. 2021. *Sequential Quantum Gate Decomposer (SQUANDER).* DOI: https://doi.org/10.5281/zenodo.4508680

[59] Leo Rogers. 2021. *The Synthesis of Nearest Neighbour Compliant Quantum Circuits*. Ph.D. Dissertation. Queen's University Belfast.

[60] Neil J. Ross and Peter Selinger. 2016. Optimal ancilla-free Clifford + T approximation of z-rotations. *Quantum Information & Computation* 16, 11–12 (2016), 901–953.

[61] P. Sarnak. 2015. Letter to Scott Aaronson and Andy Pollington on the Solovay-Kitaev Theorem and Golden Gates.

[62] Peter Selinger. 2015. Generators and relations for n-qubit Clifford operators. *Logical Methods in Computer Science* 11 (2015).

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

[64] Vivek V. Shende and Igor L. Markov. 2009. On the CNOT-Cost of TOFFOLI gates. *Quantum Information & Computation* 9, 5 (may 2009), 461–486.

[65] Vivek V. Shende, Igor L. Markov, and Stephen S. Bullock. 2004. Minimal universal two-qubit controlled-NOT-based circuits. *Physical Review A* 69, 6 (Jun 2004), 062321. DOI: https://doi.org/10.1103/PhysRevA.69.062321

[66] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2013. A sparse-group lasso. *Journal of Computational and Graphical Statistics* 22, 2 (2013), 231–245.

[67] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. 2020. t|ket⟩: A retargetable compiler for NISQ devices. *Quantum Science and Technology* 6, 1 (nov 2020), 014003.

[68] Bochen Tan and Jason Cong. 2020. Optimal layout synthesis for quantum computing. In *Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design*. IEEE, 1–9.

[69] B. Tan and J. Cong. 2020. Optimality study of existing quantum computing layout synthesis tools. In *Proceedings of theIEEE/ACM International Conference On Computer Aided Design.* (2020), 1–9.

[70] Farrokh Vatan and Colin Williams. 2004. Optimal quantum circuits for general two-qubit gates. *Physical Review A* 69, 3 (2004), 032315.

[71] Bujiao Wu, Xiaoyu He, Shuai Yang, Lifu Shou, Guojing Tian, Jialin Zhang, and Xiaoming Sun. 2019. Optimization of CNOT circuits on topological superconducting processors. arXiv:1910.14478. Retrieved 11 November 2021 from https://arxiv.org/abs/1910.14478.

[72] Ed Younis, Koushik Sen, Katherine Yelick, and Costin Iancu. 2020. QFAST: Quantum synthesis using a hierarchical continuous circuit space. arXiv:2003.04462. Retrieved 11 November 2021 from https://arxiv.org/abs/2003.04462.

[73] Ed Younis, Koushik Sen, Katherine Yelick, and Costin Iancu. 2021. QFAST: Conflating search and numerical optimization for scalable quantum circuit synthesis. arXiv:2103.07093. Retrieved 11 November 2021 from https://arxiv.org/abs/2103.07093.

[74] Jun Zhang, Jiri Vala, Shankar Sastry, and K. Birgitta Whaley. 2004. Minimum construction of two-qubit quantum operations. *Physical Review Letters* 93, 2 (Jul 2004), 020502.

[75] A. Zulehner, A. Paler, and R. Wille. 2019. Retrieved 11 November 2021 from https://github.com/iic-jku/ibm_qx_mapping.

[76] A. Zulehner, A. Paler, and R. Wille. 2019. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 7 (2019), 1226–1236.

[77] Alwin Zulehner and Robert Wille. 2019. Compiling SU(4) quantum circuits to IBM QX architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (Tokyo, Japan). 185–190.

Received March 2021; revised November 2021; accepted December 2021