IOP Publishing New J. Phys. 25 (2023) 073018 https://doi.org/10.1088/1367-2630/ace077

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
9 August 2022

REVISED
31 March 2023

ACCEPTED FOR PUBLICATION
21 June 2023

PUBLISHED
31 July 2023

PAPER

Exploring *ab initio* machine synthesis of quantum circuits

Richard Meister[1,*], **Cica Gustiani**[1] and **Simon C Benjamin**[1,2]

[1] Department of Materials, University of Oxford, Oxford OX1 3PH, United Kingdom
[2] Quantum Motion, 9 Sterling Way, London N7 9HJ, United Kingdom
[*] Author to whom any correspondence should be addressed.

E-mail: richard.meister@spc.ox.ac.uk

Keywords: quantum computing, quantum compiling, quantum circuits, variational quantum algorithms

Abstract

Gate-level quantum circuits are often derived manually from higher level algorithms. While this suffices for small implementations and demonstrations, ultimately automatic circuit design will be required to realise complex algorithms using hardware-specific operations and connectivity. Therefore, *ab initio* creation of circuits within a machine, either a classical computer or a hybrid quantum–classical device, is of key importance. We explore a range of established and novel techniques for the synthesis of new circuit structures, the optimisation of parameterised circuits, and the efficient removal of low-value gates via the quantum geometric tensor. Using these techniques we tackle the tasks of automatic encoding of unitary processes and translation (recompilation) of a circuit from one form to another. Using emulated quantum computers with various noise-free gate sets we provide simple examples involving up to 10 qubits, corresponding to 20 qubits in the augmented space we use. Further applications of specific relevance to chemistry modelling are considered in a sister paper, 'Exploiting subspace constraints and *ab initio* variational methods for quantum chemistry'. The emulation environments used were `QuEST`, `QuESTlink` and `pyQuEST`. All resources will be made openly accessible and are currently available upon request.

1. Introduction

The standard formalism for describing instructions on quantum computers is that of circuits consisting of quantum gates [1–4]. This description of reversible logic functions serves as a platform on which the capabilities of a given quantum hardware can be described as sets of available *native* gates, as well as a language in which to describe quantum algorithms using well-established universal sets of gates [5].

Native gate sets for quantum hardware often consist of a limited number of gates, which express hardware constraints like restricted qubit connectivity or a specific set of supported gates, as is the case e.g. for trapped ions [6–10], superconducting qubits [11–14], and silicon-based hardware [15–19]. Quantum algorithms, on the other hand, are often prescribed in a gate set that fits the properties of the computation [20]. This mismatch in expressibility necessitates methods for *circuit compilation*, where the description in one gate set is translated to an equivalent circuit using a different set, or, more generally, *circuit synthesis*, where a circuit is constructed from a generic description of its unitary action. This process is analogous to the compilation process on classical hardware, where high-level programming language instructions must be translated to assembly and ultimately to byte code compatible with the specific hardware it will be executed on.

Several techniques have been proposed to approach such quantum compilation tasks. *Exact unitary decompositions* apply transformations to the target unitary, which yields a solvable relationship between matrix elements of the unitary and gates in the resulting circuit [21–24]. These methods always lead to exact—but sometimes impractically long—expressions of unitaries as gate sequences. Small problems acting on only a few qubits can also be solved using exhaustive search of all possible gate sequences [25] to find the optimal circuit implementing the desired unitary. Where exhaustive search would be prohibitively costly, a tree-like structure may be used to represent circuit layouts, which can then be searched more efficiently using an approximate A* algorithm to find near-optimal solutions for problems on up to four qubits [26].

Employing various techniques to reduce the size of the considered search space—which may impact the solution optimality—this method can be used on up to six qubits [27].

For larger problems where an (impractically deep) circuit implementation is already known, the number of gates can often be reduced by partitioning the structure into smaller sub-problems. These sub-circuits can then be compiled individually, either with high precision [28] or only approximately [29].

In some cases, a known circuit for the desired quantum algorithm may be incompatible with the connectivity constraints of the hardware. This task is often called *qubit routing* or *circuit transformation*, and several methods have been applied to efficiently find the necessary SWAP operations between gates, including simulated annealing [30], tabu search [31], artificial neural networks [32, 33], and specialised heuristics [34–37]. Furthermore, sometimes SWAPs can be avoided altogether [38].

A different approach is to dynamically construct a circuit through machine learning, heuristics, or metaheuristics. Several techniques have been proposed, including pseudorandom walks [39], genetic algorithms [40], temporal planning [41], and deep learning [42]. Such methods can sometimes also be used to synthesise approximations to a desired unitary, rather than a perfect re-expression, resulting in shorter circuits than exact compilation. Of particular relevance for the present work are approaches which randomly propose circuit structure changes to try and minimise some cost function. Examples of this are given in the [43–45]. Each of these works proposes the use of different cost functions, but all of them employ Metropolis-Hastings random sampling to minimise the cost.

It is also noteworthy that while many variational quantum eigensolvers (VQEs)—which seek to generate a circuit to prepare the ground state of a Hamiltonian from a given input state—use fixed circuit structures, there have been advancements in generating gate sequences dynamically, for example ADAPT-VQE [46, 47] and the evolutionary VQE (EVQE) [48]. As we will discuss, these approaches can, in principle, also be used for circuit synthesis tasks.

The present manuscript presents a broad exploration of the efficacy of variational quantum algorithm-based methods for circuit synthesis and (re-)compilation. We leverage previously described methods as well as techniques that are, to the authors' best knowledge, novel. To assess the capabilities of each approach we use emulated quantum computers, and document the results of systematic studies across a variety of contexts (gates sets, connectivities, etc) and problem scales (up to 20 qubits). Based on the trends we observe, we draw conclusions on the prospects and value of the approach. We note that small-scale circuit manipulation is certainly practical and can be of key significance in realising components of larger circuits. However, the direct synthesis of entire circuits at a scale relevant to quantum advantage does remain a challenging task whose feasibility is unclear.

To the best of the authors' knowledge, the novel contributions of this work are the restriction of the cost function to a specific subspace (section 2.3), detection of redundant gates using the quantum metric tensor (QMT) (section 3.4.2), and the application of the adapted versions of random search (section 3.6.2) and tabu search (section 3.6.3) to the problem of finding the circuit structure. We selected these methods in order to explore the efficiency with which substantial circuits can by synthesised using these specific (semi-)random ways to traverse the circuit structure space, which has not previously been discussed in the literature.

The rest of this paper is structured as follows. In section 2 we describe the cost function we use to quantify the closeness of a proposed circuit to the target unitary, which heavily draws ideas from [45] and [49]. Section 3 describes the routines and subroutines we use to generate circuits in detail. We then apply these routines to various problems, report the results in section 4, and discuss their implications as well as potential future improvements in section 5.

2. Unitary equivalence via energy minimisation

2.1. Previous work

The present manuscript substantially builds upon earlier formalisms introduced in, for example, [45, 49]. In this section, we briefly recapitulate their methods and motivate our extensions to it.

When synthesising a circuit $\mathcal{C}$ for a given unitary U, some measure of how well $\mathcal{C}$ approximates U is necessary to drive an optimisation routine towards (approximate) equivalence of $\mathcal{C}$ and U. Jones and Benjamin [49] use the energy of an artificial Hamiltonian to provide such a measure for the case where equivalence is wanted only for a single input state $|\psi_0\rangle$. Khatri *et al* [45], on the other hand, provide a method called *Hilbert–Schmidt test*, which uses the average fidelity $\langle\psi|\mathcal{C}^\dagger U|\psi\rangle$ over Haar-distributed random states $|\psi\rangle$ to take *all* input states into account when measuring the closeness of U and $\mathcal{C}$. In the following, we start from the formalism in [49] and extend it to arrive at a variant of the Hilbert–Schmidt test, which we will use as our cost function.

The technique in [49] starts by first applying the target unitary U to the target input state $|\psi_0\rangle$. Then, the task is to find a circuit $\mathcal{C}^\dagger$ that inverts the action of U, such that at the output $|\psi_1\rangle := \mathcal{C}^\dagger U|\psi_0\rangle$ the initial state

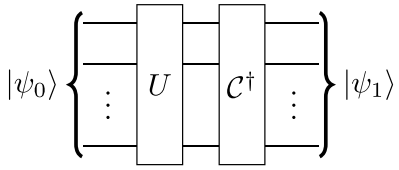

Figure 1. The setup used in [49] to synthesise a circuit $\mathcal{C}$ which has the same action on $|\psi_0\rangle$ as U. The target is $|\psi_1\rangle \sim |\psi_0\rangle$.

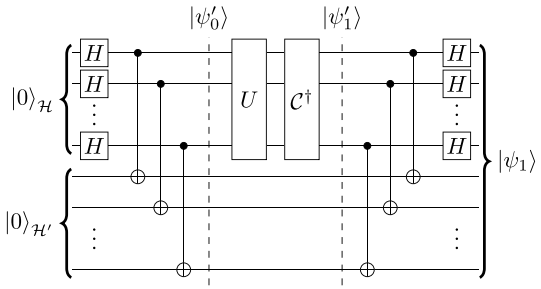

Figure 2. The circuit setup equivalent to a Hilbert–Schmidt test in [45], which we used to synthesise $\mathcal{C} \sim U$. The target is $|\psi_1\rangle \sim |0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}'}$.

$|\psi_0\rangle$ is recovered up to a global phase[3]. If the output is proportional to the input, $|\psi_0\rangle \sim |\psi_1\rangle$, then necessarily $\mathcal{C} \sim U$ holds for the input state $|\psi_0\rangle$. Using an appropriately constructed gapped Hamiltonian $\tilde{H}$ whose ground state is $|\psi_0\rangle$, the initial problem is now an energy minimisation task—as depicted in figure 1—of the form

$$\min_{\mathcal{C}} \langle\psi_0|U^\dagger \mathcal{C}\tilde{H}\mathcal{C}^\dagger U|\psi_0\rangle. \tag{1}$$

However, it is often desirable to synthesise a circuit $\mathcal{C}$ that completely recovers the action of a given unitary U for *all* relevant input states. This might be the full Hilbert space of $|\psi\rangle$—we call it $\mathcal{H}$—or a closed subspace thereof, depending on the application. Recent research [44, 50–53] has shown that in some cases, using a subset of $k \ll \dim(\mathcal{H})$ random states $|\psi_k\rangle$ sampled from $\mathcal{H}$, and minimising the sum of their energies

$$\min_{\mathcal{C}} \sum_k \langle\psi_k|U^\dagger \mathcal{C}\tilde{H}\mathcal{C}^\dagger U|\psi_k\rangle \tag{2}$$

is a sufficient condition to get unitary equivalence. However, this only holds true for highly structured operators. Therefore, in this work, we use cost functions building on [45] to probe all states in the Hilbert space simultaneously, rather than restricting to a random sample thereof.

Starting from equation (1) and figure 1, it is possible to achieve full unitary equivalence of U and $\mathcal{C}$ by exploiting the Choi–Jamiołkowski isomorphism [54, 55]. From an all-zero input we first create the maximally entangled state $|\psi_0'\rangle = \sum_k |k\rangle \otimes |k\rangle$. We can then map the action of $\mathcal{C}^\dagger U$ on every state in $\mathcal{H}$ to the action of $\mathcal{C}^\dagger U \otimes \mathbb{1}$ on the single state $|\psi_0'\rangle$ in the space $\mathcal{H} \otimes \mathcal{H}'$, where $\mathcal{H}'$ is a copy of $\mathcal{H}$. Figure 2 shows a circuit construction of this method, which is equivalent to the Hilbert–Schmidt test introduced in [45]. The output state $|\psi_1'\rangle$ is proportional to the input state $|\psi_0'\rangle$ if and only if $\mathcal{C} \sim U$. Therefore, applying the inverse of the preparation circuit that created $|\psi_0'\rangle$ returns the state to the original computational all-zero state if indeed $\mathcal{C} \sim U$. In order to estimate how close we are to this ideal, we can use any Hamiltonian $\tilde{H}$ which has $|0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}'}$ as its unique and gapped ground state as an artificial Hamiltonian. Because we know an expectation value of $\langle\psi_1|\tilde{H}|\psi_1\rangle = 0$ exactly corresponds to $U \sim \mathcal{C}$, energy minimisation techniques can be used to find a circuit $\mathcal{C}$ which is equivalent to a given unitary U for all input states. Note that for $\tilde{H} = |0\rangle\langle 0|_{\mathcal{H}} \otimes |0\rangle\langle 0|_{\mathcal{H}'}$, we exactly recover the Hilbert–Schmidt test with a target expectation value of $\langle\tilde{H}\rangle = 1$.

Figure 3 in section 2.3 shows a—to the authors' best knowledge—novel modification to this setup, which interpolates between figures 1 and 2 by considering a specific subspace of $\mathcal{H}$, on which we elaborate in section 2.3.

[3] In this work we use $\sim$ between operators to denote equivalence up to a global phase, i.e. $A \sim B \leftrightarrow A = e^{i\theta}B$.

As shorthand notation for our cost function we will write $\langle \tilde{H} \rangle$ to mean the expectation value $\langle \psi_1 | \tilde{H} | \psi_1 \rangle$ in the full augmented space with $|\psi_1\rangle$ as produced by the circuit in figure 2. Possible choices for $\tilde{H}$ are discussed in the next subsection.

Using $\langle \tilde{H} \rangle$ to determine unitary equivalence ignores global phase factors by which U and $\mathcal{C}$ might differ. In most scenarios, this is desirable, since such a global phase is physically irrelevant. However, if the resulting $\mathcal{C}$ is to be used in a controlled fashion within a larger circuit, a global phase mismatch of U and $\mathcal{C}$ becomes a physically relevant relative phase. In this case, an additional phase gate with an appropriate parameter must be added to $\mathcal{C}$ at the end of the synthesis process.

2.2. Synthesis Hamiltonians

When expressing the condition of unitary equivalence as an energy minimisation problem, there is some freedom in choosing an appropriate $\tilde{H}$, as any gapped Hamiltonian with the all-zero computational basis state as its ground state can be used. In this work, we consider the two Hamiltonians

$$H_{\text{sum}} = \frac{1}{N} \sum_k \sigma_k^z \tag{3}$$

where σ_k^z is the Pauli-Z operator acting on qubit k, and N is the total number of qubits, and

$$H_{\text{proj}} = \mathbb{1} - |0\rangle\langle 0|. \tag{4}$$

While H_{proj} corresponds—as previously mentioned—to the Hilbert–Schmidt test, H_{sum} is more closely related to the *local* Hilbert–Schmidt test [45], and the local cost function proposed in [56]. Both of these Hamiltonians have a ground state energy of 0 and maximum energy of 1, they differ in the energy structure of their excited states, and thus also in how they measure the closeness of a circuit is to a target unitary.

The projector-based H_{proj} simply measures the overlap with the desired all-zero state, while assigning every other product state the same maximum energy of 1.

On the other hand, the sum-based H_{sum} assigns each computational basis state an energy proportional to its Hamming distance [57] from the ground state, i.e. it measures how many bit flips would be necessary to get to the desired all-zero state. The state $|0\ldots01\rangle$ therefore has a lower energy than the state $|1\ldots11\rangle$ when measured with H_{sum}, while H_{proj} evaluates them as being at equal distances from the target.

The choice of Hamiltonian can also have a significant impact on the difficulty of finding the optimal parameters. For instance, [45, 56], discuss that global cost functions—like H_{proj}—are very likely to come across barren plateaus during the parameter optimisation, while employing local costs—such as H_{sum}—is much less prone to such problems.

Even though in this work we only consider H_{sum} and H_{proj}, many other Hamiltonians may be used. Depending on the application, they can be designed to emphasise different properties of what constitutes a *good* output state, and may penalise some highly undesirable properties more harshly than others.

2.3. Subspace compilation

For some tasks, synthesising the entire unitary is unnecessary. For instance, some chemistry problems demand symmetry conservation, such as particle number preservation and spin preservation. If a unitary with such a structure is applied to a state whose entire weight is within one such subspace, the operation outside that subspace becomes irrelevant. In this task, the unitary U has a block-diagonal form,

$$U = U_1 \oplus U_2 \oplus U_3 \oplus \cdots \oplus U_n, \tag{5}$$

where U_j is a unitary and $\oplus$ means the direct sum. The task now is discovering a circuit that correctly implements the unitary U_j, given the entire unitary U.

We address the compilation within such a subspace as *subspace compilation*. Assuming that the subspace size is much smaller than the full space, we will demonstrate how our learning method considerably improves the synthesis time and decreases the number of gates in the final circuit compared to the compilation in the full space.

Consider a block-diagonal unitary U acting in a Hilbert space $\mathcal{H}$ of dimension $d = 2^n$. Without loss of generality, suppose $U = U_1 \oplus U_2$, i.e. a block-diagonal unitary comprised of two blocks. Let U_1 be an operator in $\mathcal{H}_1$ and U_2 an operator in $\mathcal{H}_2$, the unitary U acts on $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2$, where $\mathcal{H}_1$ and $\mathcal{H}_2$ are subspaces of $\mathcal{H}$. Given that H_1 has dimension d_1 and H_2 has dimension d_2, the dimension of $\mathcal{H}$ is $d_1 + d_2 = d$.

Given access to U, compiling U_1 on subspace $\mathcal{H}_1$ using our *ab initio* technique has a cost evaluation setup shown in figure 3; this circuit generalises the compilation explained in section 2.2.

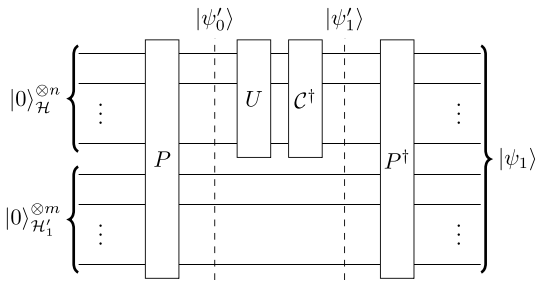

Figure 3. One cost evaluation in a subspace compilation. The unitary P prepares a maximally entangled state $|\Phi\rangle_{\mathcal{H}\mathcal{H}_1'}$ according to equation (6).

First, we prepare two quantum registers: the main register (representing $\mathcal{H}$) with n qubits and the ancilla register (representing $\mathcal{H}_1'$) with $m = \lceil \log_2 d_1 \rceil$ qubits. Second, we prepare a maximally entangled state $|\Phi\rangle_{\mathcal{H}\mathcal{H}_1'}$ on both registers $\mathcal{H} \otimes \mathcal{H}_1'$, where

$$P|0\rangle_{\mathcal{H}\mathcal{H}_1'} = |\Phi\rangle_{\mathcal{H}\mathcal{H}_1'} = \frac{1}{\sqrt{d_1}} \sum_{s_j \in S} |s_j\rangle_{\mathcal{H}} |j\rangle_{\mathcal{H}_1'}, \tag{6}$$

with a unitary operator P, and S, an orthonormal basis spanning $\mathcal{H}_1$. Note that $|\Phi\rangle_{\mathcal{H}\mathcal{H}_1'}$ is maximally entangled between spaces $\mathcal{H}$ and $\mathcal{H}_1'$, i.e. $\mathrm{Tr}_{\mathcal{H}}(|\Phi\rangle\langle\Phi|)_{\mathcal{H}\mathcal{H}_1'} = \mathbb{1}_{\mathcal{H}_1'}$ and $\mathrm{Tr}_{\mathcal{H}_1'}(|\Phi\rangle\langle\Phi|)_{\mathcal{H}\mathcal{H}_1'} = \mathbb{1}_{\mathcal{H}}$.

3. *Ab initio* circuit synthesis

In this section we discuss algorithms and subroutines used to vary the structure of ansatz-circuits and their parameters in order to minimise the expected energy under specific Hamiltonians, thereby solving various circuit synthesis and VQE problems. We introduce notation and definitions for our synthesis protocols momentarily, while sections 3.2–3.5 describe subroutines which are then used within the algorithms laid out in section 3.6.

3.1. Formalism

The building blocks of our circuits are primitive gates G_k, which can be single-qubit rotations, multi-qubit rotations, SWAPs, etc acting on different sets of qubits. We assume that every gate has a classical parameter, e.g. a rotation angle, associated with it. The formalism is easily extended to also include non-parametrised gates; they are simply gates whose parameter is permanently fixed. In our terminology, altering the parameter associated with a gate does not constitute a replacement of the gate itself.

We refer to a specific set of gates $\mathcal{L} = \{G_k\}$ as a *gate library*. Note that each G_k specifies which qubits the gate acts upon; for example, the Pauli-X rotations on different qubits 1 and 2—R_1^x and R_2^x respectively—would be considered two separate gates G_k. This allows a gate library to include information about only locally available gates and qubit connectivity.

A *circuit structure*—or ansatz—$\mathcal{C}$ can be represented by an ordered sequence of such primitive gates

$$\mathcal{C} := (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{N-1}), \tag{7}$$

where $\mathcal{C}_k \in \mathcal{L}$.

Before the circuit can be applied to a quantum state, a parameter vector $\underline{\theta}$ containing the parameter θ_k for each gate $\mathcal{C}_k$ must be assigned. We write

$$\mathcal{C}(\underline{\theta}) := (\mathcal{C}_0(\theta_0), \mathcal{C}_1(\theta_1), \ldots, \mathcal{C}_{N-1}(\theta_{N-1})).$$

Applying a circuit $\mathcal{C}(\underline{\theta})$ to a state $|\psi\rangle$ means evaluating

$$\mathcal{C}_{N-1}(\theta_{N-1}) \cdot \ldots \cdot \mathcal{C}_1(\theta_1) \cdot \mathcal{C}_0(\theta_0) |\psi\rangle.$$

A given circuit $\mathcal{C}(\underline{\theta})$ can be modified in two fundamentally different ways. One is to change the parameters $\underline{\theta}$, which we call *parameter optimisation*. The other is to add or remove gates to or from the circuit structure $\mathcal{C}$, which we refer to as *circuit structure modifications*. The procedures for how we perform these modifications are explained in detail in the following subsections. Our routines are also given as simplified versions in high-level pseudocode in appendix A.2, which might miss some performance enhancing tweaks for the sake of clarity and brevity.

3.2. Parameter optimisation

For a given circuit structure $\mathcal{C}$ containing parameterised gates—often called *ansatz circuit*—we want to find the parameter vector $\underline{\theta}$ which minimises the expected energy of our artificial Hamiltonian $\tilde{H}$, i.e. the cost function. At this minimum, the circuit $\mathcal{C}(\underline{\theta})$ most closely approximates the desired unitary U within the scope of its parameter space. In our algorithms, we use imaginary time evolution [58, 59] in a slightly modified version. We first compute the matrix object[4]

$$A_{ij} = \mathrm{Re}(\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle), \tag{8}$$

which we refer to as the QMT [60–62], and the *gradient vector*

$$B_i = -\langle\partial_i\psi|H|\psi\rangle. \tag{9}$$

The time evolution of the parameter vector $\underline{\theta}$ is then given by [59]

$$\mathbf{A}\dot{\underline{\theta}} = \underline{B}. \tag{10}$$

Using the forward Euler method [63] we thus get the update rule for the parameter vector

$$\underline{\theta}_{t+1} = \underline{\theta}_t - \lambda\mathbf{A}^{-1}\underline{B}. \tag{11}$$

The matrix $\mathbf{A}$ is often close to singular, so some regularisation method is required to stabilise the iteration. We use Tikhonov regularisation, but other methods may be used as well.

To find a suitable λ, we use an idea similar to the one presented in [49]. In each iteration, we start from a small (arbitrary) initial value of $\lambda_0 = 0.05$ for each time step. But, because the evaluation of the QMT and the gradient vector may be expensive operations [64] compared to the evaluation of the expected energy for a given set of parameters, we then try to exponentially increase the step size until we find a local minimum along the established step direction. This means repeatedly multiplying λ by some constant factor κ until an energy increase is found, and then accepting the immediately preceding λ-value. However, if the initial step size turns out to already increase the energy, we instead shrink λ exponentially until we find a decrease in energy or hit a minimum step size. This exponential search is almost always useful in emulators, where the gradient direction is known to high numerical precision. Its usefulness in avoiding barren plateaus is reduced on real quantum hardware when shot noise limits how precisely the gradient direction can be determined.

To detect convergence, we use absolute and relative changes of the energy, and require one of the conditions to be met a number $k_{\mathrm{conv}} \sim 5$ of times. Algorithm 1 shows the full procedure.

3.3. Introduction of new gates

Dynamic expansion of a given ansatz circuit has been explored by VQE methods [46–48] and in more general contexts [43, 44, 50]. The formalism we introduce here is very generic, but naturally shares some ideas with previous works, especially [44].

The algorithms introduced later rely on the concept of a *move*, which refers to the most basic possible modification of a circuit structure $\mathcal{C}$, i.e. the insertion of one additional gate at some position in the gate sequence. Given a library of gates $\mathcal{L}$ and a circuit with N gates, it is convenient to define a move as a tuple (G, n), with a gate $G \in \mathcal{L}$ and an index $0 \leqslant n \leqslant N$. Applying such a move to an existing circuit $\mathcal{C}$ means inserting gate G at position n.

$$\mathcal{C} \mapsto (\mathcal{C}_0, \ldots, \mathcal{C}_{n-1}, G, \mathcal{C}_n, \ldots, \mathcal{C}_{N-1}). \tag{12}$$

The routine ApplyMove in algorithm 2 performs exactly this action.

To access the gate and the index of a move, we use subscripts G and n, respectively, e.g. if $M = (R_1^x, 4)$, then $M_G = R_1^x$ and $M_n = 4$. This definition allows us to conveniently pass around *moves* between functions, which will be useful later.

For a circuit containing N gates, all possible moves are given by

$$\mathcal{M}_{\mathrm{all}} = \{(G, n) \mid 0 \leqslant n \leqslant N \text{ and } G \in \mathcal{L}\}. \tag{13}$$

However, many of the moves in $\mathcal{M}_{\mathrm{all}}$ will lead to redundancies in the circuit, because neighbouring identical gates acting on the same qubits can be merged straightforwardly. We eliminate modifications

[4] We use the shorthand notation $|\partial_\mu\psi\rangle := \frac{\partial|\psi(\underline{\theta})\rangle}{\partial\theta_\mu}$.

leading to such obvious redundancies and generate a set containing only potentially useful moves. The specific method we used is given in algorithm 2 and works as follows. We separately look at each qubit k, and in one iteration only consider the gates in $\mathcal{C}$ acting on qubit k. Potentially useful modifications are then insertions of gates from the library between consecutive pairs of gates which also act on qubit k, but are different to the previous and following gate acting on qubit k.

In some cases—e.g. if some gates in $\mathcal{L}$ commute with one another—algorithm 2 will still include moves that lead to redundancies. The QMT as defined in section 3.2 can be used to detect such redundant gates. The details of this operation are given in the next subsection.

3.4. Removal of superfluous gates

In a circuit $\mathcal{C}$, not all gates in the sequence necessarily contribute to generating the desired unitary in a useful way. We use three distinct techniques with varying computational cost and ability to detect such redundancies, which we discuss in the following.

In our algorithms, we use these methods in the order *small parameter* → *QMT-assisted* → *trial and error*, as shown in algorithm 4. Each method can, in principle, also detect all redundancies of the previous methods[5], but is more costly to perform, which makes this staged approach useful.

3.4.1. Small parameter removal

The computationally cheapest way to detect non-contributing gates is by checking the associated parameters after optimising them[6]. For every circuit parameter close to 0 modulo[7] 2π, $\theta_k \approx 0 \bmod 2\pi$, the corresponding gate $\mathcal{C}_k$ can be removed immediately.

3.4.2. QMT assisted removal

As a more sophisticated and computationally slightly more expensive method to detect further redundancies, we check the QMT—as defined in equation (8)—for linearly dependent rows.

The QMT contains information about how the output state changes with respect to varying the parameters. If rows i and j in this tensor are linearly dependent, the linearised actions of $\mathcal{C}_i$ and $\mathcal{C}_j$ are equivalent in the tangent space of the circuit $\mathcal{C}$ at the current position $\underline{\theta}$ in parameter space. Intuitively, this means that changes to the parameters θ_i and θ_j from their current values would move the state in the same direction inside some subspace of the full Hilbert space. While this does not guarantee that $\mathcal{C}_i$ and $\mathcal{C}_j$ have equivalent actions in the full Hilbert space $\mathcal{H}$ or at a different point $\underline{\theta}'$ in parameter space, it is a strong indication of it. We therefore use

$$\left|(A_{i,\cdot}^{\intercal} \cdot A_{j,\cdot}) - \|A_{i,\cdot}\| \, \|A_{j,\cdot}\|\right| < \varepsilon_{\mathrm{QMT}}, \tag{14}$$

where $A_{i,\cdot}$ is the ith row vector of $\mathbf{A}$, as a heuristic for detecting potentially redundant gates with an appropriately small $\varepsilon_{\mathrm{QMT}}$.

At many points in the iteration the QMT is already known from the previously performed parameter optimisation and does not need to be explicitly re-calculated. As calculating this matrix is computationally relatively expensive, this saves valuable computing time.

The removal is performed as follows. For all pairs of rows i, j we check the condition in equation (14). If it is fulfilled, we adjust the parameter of the first gate to $\theta_i \leftarrow \theta_i + \theta_j$ and remove the gate $\mathcal{C}_j$ from the circuit $\mathcal{C}$ as well as θ_j from the parameter vector $\underline{\theta}$. If this removal does not significantly increase the energy, the new circuit is kept, otherwise the deletion is reverted. Among other redundancies, this method allows the relatively easy detection of identical gates separated by gate sequences they (non-trivially) commute with.

3.4.3. Trial and error removal

The above method is good at finding redundancies where one gate can absorb a different one into its parameter. It cannot, however, detect cases where multiple gates need to adjust their parameters in order to compensate the removal of one specific gate. We therefore employ a third strategy, which is computationally more expensive, but can also detect much more subtle redundancies.

Given a circuit $\mathcal{C}$ and a set of gate indices $\mathcal{R}$ to be considered candidates for removal, we delete $\mathcal{C}_k$ for each $k \in \mathcal{R}$ from the circuit without replacement, and run a full parameter optimisation as in section 3.2

[5] In the generalisation of also allowing non-parametrised gates in the circuit—which we do not discuss further—only *trial and error* may be used for those specific gates.

[6] We assume that every gate G approaches the identity for small parameters, i.e. $\lim_{\theta \to 0} G(\theta) = \mathbb{1}$.

[7] For practical reasons we use the slightly unusual definition of $a \bmod b = a - b\lfloor a/b + .5 \rfloor$ which returns values in the interval $[-b/2, b/2)$ instead of the usual $[0, b)$.

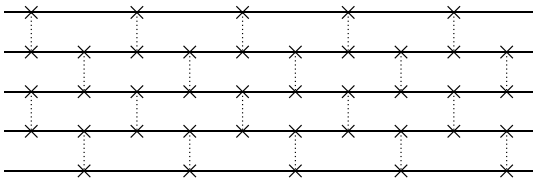

Figure 4. The initial circuit we use for gate libraries $\mathcal{L}$ where two-qubit gates are limited to nearest neighbours.

with the modified circuit. If the energy after the relaxation is not significantly higher than before, the deleted gate is considered redundant and remains removed. We refer to this method as a *hard removal*, because the circuit is abruptly taken to a different point in parameter space, potentially far away from a local minimum.

An alternative to the aforementioned *hard removal* explored by the authors but not reported here, is a method we refer to as *soft removal*. The parameter θ_k, for $k \in \mathcal{R}$, is shifted towards zero by some predetermined amount, and a single imaginary time step for all parameters except θ_k is performed right afterwards. This procedure is repeated until the parameter is close to zero, and only then is the gate completely removed. This allows the circuit to stay close to the local minimum it is already in. Therefore this method can occasionally lead to better results.

3.5. Initial circuit

When the gates in the considered library $\mathcal{L}$ have limited connectivity, i.e. not every qubit can interact with every other qubit directly, finding useful gate additions by randomly adding gates can become increasingly improbable. For example, in a linear chain of qubits with nearest-neighbour connectivity, having qubit 0 interact with qubit 3 requires the correct simultaneous addition of at least three gates. The effort of finding the correct combination of three gates is further hampered by the fact that—at least for the artificial Hamiltonians $\tilde{H}$ we use for our circuit synthesis tasks—the energy landscape with regard to adding only a subset of those gates is flat.

To alleviate these connectivity limitations, it can be helpful to choose not to start with an empty circuit, but to have an initial structure of parameterised SWAP gates, where every qubit can be close to every other qubit at some point, and is able to—but does not need to—subsequently return to its original position. This can be achieved by an ansatz of the form

$$\mathcal{C}^{(0)} = \prod_{n=0}^{N} \left[\prod_{k=1}^{\lceil N/2-1 \rceil} E_{2k,2k+1} \prod_{k=0}^{\lfloor N/2-1 \rfloor} E_{2k,2k+1} \right] \tag{15}$$

where N is the number of qubits and

$$E_{i,j} := \exp\left(i \frac{\theta}{2} \mathrm{SWAP}_{i,j} \right) \tag{16}$$

is a gate which performs no action for $\theta = 0$ and swaps the qubits i and j if $\theta = \pi$. The ansatz is visualised in figure 4, where the E gate is indicated by the usual SWAP symbol, but drawn with dotted lines.

The $E_{i,j}$ gates are typically not part of the gate library $\mathcal{L}$ and must be compiled to it separately. If all parameters converge to either 0 or π, it is sufficient to compile the SWAP gate to the desired gate library $\mathcal{L}$.

3.6. Circuit structure finding

Here we describe the algorithms we used to synthesise quantum circuits, all of which are adaptions of well-known optimisation techniques.

3.6.1. Hill climbing

The simplest and most straightforward algorithm we employ is a variant of hill climbing [65], which iteratively searches some *neighbourhood* of the current circuit structure, and accepts the lowest energy solution within that neighbourhood. In our case, we define the neighbourhood of a circuit structure as all circuits reachable by applying N_{moves} *moves* as defined in section 3.3. For hill climbing, we only consider $N_{\mathrm{moves}} = 1$.

Starting from some initial circuit $\mathcal{C}^{(0)}$, all moves are tried, their parameters optimised according to section 3.2, and the energies of the resulting circuits are recorded. The move which resulted in the lowest energy is kept and becomes the new circuit $\mathcal{C}^{(1)}$. This procedure is repeated until the energy falls below a

given threshold. Removal of non-contributing gates is only done once at the end of the iteration. Algorithm 5 shows this procedure.

Because it only checks the immediate neighbourhood of the current circuit for improvements, this method quickly gets stuck in local minima of the cost function. This problem is typical for hill climbing algorithms. In some very limited cases, it is possible to extend the search radius to all combinations of two or more sequential steps, i.e. $N_{\text{moves}} > 1$, in order to escape such local optima. However, in most cases the search space volume grows very rapidly with the search depth, making larger search radii impractical.

3.6.2. Random search

To be able to escape local minima in which hill climbing gets stuck, we must make larger steps in configuration space. However, as mentioned in the previous section, the size of the neighbourhood grows too rapidly to exhaustively search it. We therefore resort to a variant of random search [65], which in each iteration proposes a random modification to the circuit, and accepts it if it lowers the energy.

A single proposed modification—we will call this a *random step*—consists of applying a number of N_{moves} randomly chosen *moves* to the circuit, and only then optimising its parameters. This essentially means adding N_{moves} random gates to the circuit. Such a random step takes us further in the space of circuit structures and thus has the ability to escape from local optima. Because many of the added gates are potentially not contributing to the reduction in energy, we attempt to remove as many of the newly added gates as possible after each random step via the methods discussed in section 3.4.

When drawing a random move, some gate types (e.g. controlled rotations) might be overrepresented compared to others (e.g. local rotations), simply because there are more of them. This also skews the number of gates by type in the final circuit, which we found to sometimes hinder performance. We therefore first sort all moves into two groups, one containing only moves with single-qubit gates, and the other containing only those with two-qubit gates. To draw a random move, we first choose one of the groups with equal probability, and then uniformly draw a move from the chosen group. For clarity we omit this detail in algorithm 3.

Instead of simply accepting every random step that lowers the cost function, we furthermore found it beneficial to sample a small number $N_{\text{samp}} \sim 10$ of random steps and choose to keep only the step resulting in the lowest cost. Algorithm 3 shows this procedure.

3.6.3. Tabu search

As an extension to random search—which relies purely on chance to find useful modifications to the circuit—we also used a variant of tabu search [66] to avoid repeated application of unsuccessful circuit modifications. Here we briefly outline the idea behind the algorithm and its potential pitfalls.

The overall structure of our variant of tabu search is exactly the same as for random search in algorithm 3, but with $N_{\text{samp}} = 1$. It has however, the additional feature of a *tabu list*, which we call τ. Whenever a move is performed, it is recorded into this list, together with a label recording at which iteration the move was performed. For a number of iterations $T_{\text{tabu}} \sim 20$, the same move must then not be repeated. Therefore, before choosing a random move, all tabu moves are removed from the set of potential moves.

To keep the action of the moves consistent when positions in the circuit change due to the insertion or deletion of gates, we update the insertion indices $M_n\ \forall M \in \tau$ in the stored moves accordingly, whenever the circuit changes. This means decrementing all stored indices $M_n > m\ \forall M \in \tau$ by 1 if the gate with index m is deleted, and incrementing all indices $M_n \geqslant m\ \forall m \in \tau$ by 1 when a new gate at index m is inserted.

4. Results

We present our numerical findings for different applications of our method in this section. Unless otherwise noted, the calculations use the default hyperparameters listed in table A1 in appendix A.1. The complete set of data generated for this section is available at reference [67].

4.1. Cost functions as proxy for unitary equivalence

To determine whether two operators[8] $\mathcal{C}$ and U are equivalent, an appropriate metric to use is the global-phase invariant operator norm of their difference

$$\mathcal{D}(U,\mathcal{C}) = \min_{\phi} \|U - e^{i\phi}\mathcal{C}\| = \min_{\phi} \left[\max_{|\psi\rangle} \|U|\psi\rangle - e^{i\phi}\mathcal{C}|\psi\rangle\|_2\right], \tag{17}$$

which we will refer to as the *operator distance*. It is the maximum L_2-norm of the difference (and thus the Euclidean distance) between the desired output $U|\psi\rangle$ and the output of the recompiled version $\mathcal{C}|\psi\rangle$.

[8] From here on we will omit explicit indication of the parameter vector $\underline{\theta}$ wherever practical.

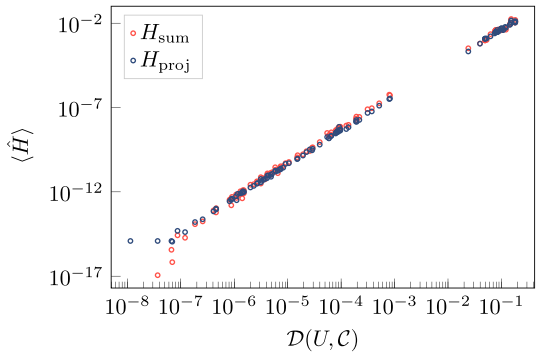

Figure 5. Operator distance between the target U and recompiled $\mathcal{C}$ unitaries for a set of recompiled QFT circuits on 5 qubits. The data points at very small costs are in the range of numerical noise and likely influenced by finite data type accuracy.

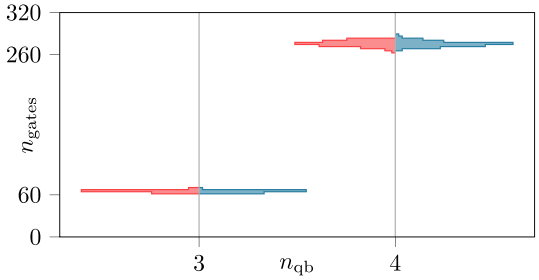

Figure 6. Histograms of the number of gates required to express Haar-distributed unitaries as a circuit containing only gates from $\mathcal{L}_{\text{allrot}}$ for different numbers of qubits. Left (red) data was obtained using tabu search, right (blue) data is from our random search algorithm.

For the calculations only considering a subspace, we also define

$$\mathcal{D}_{\mathcal{S}}(U, \mathcal{C}) := \mathcal{D}(\Pi_{\mathcal{S}} U \Pi_{\mathcal{S}}, \Pi_{\mathcal{S}} \mathcal{C} \Pi_{\mathcal{S}}) \tag{18}$$

where $\Pi_{\mathcal{S}}$ is the projector onto the relevant subspace.

Throughout our results we use the expected energy of H_{sum} or H_{proj} as defined in equations (3) and (4) to assess how closely a constructed circuit $\mathcal{C}$ reproduces the desired unitary U. Note, however, that in general $\langle \tilde{H} \rangle \nsim \mathcal{D}(\mathcal{C}, U)$. This is easily demonstrated with an n-controlled Pauli-Z gate $C_{0..n-1}[\sigma_n^z]$, for which the identity operator yields an energy of $\langle H_{\text{proj}} \rangle = \langle H_{\text{sum}} \rangle = 2^{-n}$, but the operator distance has a macroscopic value of $\mathcal{D}(\mathbb{1}, C_{0..n-1}[\sigma_n^z]) = \sqrt{2}$.

We therefore surveyed the resulting circuits of many 5-qubit QFT synthesis results and compared the cost functions of $\langle H_{\text{sum}} \rangle$ and $\langle H_{\text{proj}} \rangle$ to the actual operator distance $\mathcal{D}$. The results are plotted in figure 5 and show that close to convergence the used cost functions are both very good proxies for the actual operator distance. We therefore confidently use our cost functions to assess the quality of the recompiled circuit.

4.2. Random unitaries

4.2.1. Dense unitaries

To test and benchmark our circuit synthesis protocols, we generated uniformly Haar-distributed random unitaries[9] as targets, and tried to find circuit representations for them. For each target on N qubits we ran a single attempt to create a circuit performing the same action using the gate set of

$$\mathcal{L}_{\text{allrot}} = \{R_k^\sigma, C_\ell[R_k^\sigma] \mid \sigma \in \{x, y, z\}, k, \ell \in [N] \text{ and } k \neq \ell\} \tag{19}$$

where $[N] \equiv \{1, \ldots, N\}$, which contains all local single-qubit Pauli rotations R_k^x, R_k^y, and R_k^z on each qubit k, as well as single-controlled versions thereof with no connectivity constraints. The results of 100 random unitaries per method and number of qubits are shown in figure 6.

[9] The unitaries were created using `scipy.stats.unitary_group` in Python and `CircularUnitaryMatrixDistribution` in Mathematica.

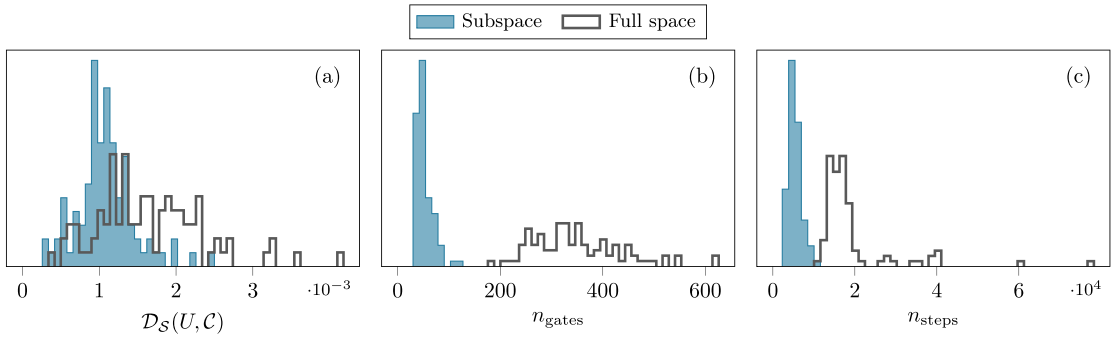

Figure 7. Normalised histograms showing properties of synthesised circuits $\mathcal{C}$ recovering the action of single 4-state block U_1 within a block-diagonal operator U on 4 qubits. The blocks making up U are each Haar distributed random unitaries. Empty histograms in grey show results for the full space method, filled histograms in blue show subspace results. Circuits for same unitary U were synthesised 100 times. (a) Operator distance as in equation (17) within the subspace $\mathcal{H}_1$. (b) Number of gates in the resulting circuit. (c) Number of total imaginary time evolution steps needed by the algorithm.

We find that the number of gates in the circuits produced by random and tabu search correlates well with the increasing degrees of freedom of the targets, which for a dense unitary on N qubits is 2^{2N}. Therefore, for 3 qubits, we would on average expect to require no fewer than 64 gates, and find a mean number of 65 gates in our synthesised circuits. For 4 qubits, we observe a mean of 275 gates, where at least 256 would be expected. This indicates that, at least for dense, unstructured unitaries, the circuits produced by tabu and random search do not contain a large number of superfluous gates.

Note that the results show no significant difference between tabu and random search, which we discuss in section 4.3.

4.2.2. Subspace compilation

In order to assess the efficacy of compiling a unitary only in a certain subspace, we generated a random unitary operator which is block-diagonal in the computational basis when sorted by Hamming weight, i.e. the number of ones in each state belonging to the same block is equal. Each of these blocks was then assigned a Haar-distributed random unitary. We synthesised circuits representing the full unitary, as well as only the block with a Hamming weight of 1, i.e. the states $\{|0001\rangle, |0010\rangle, |0100\rangle, |1000\rangle\}$, using random search. For numerical reasons—and because at this problem size barren plateaus proved not to be an issue—we employed $\langle H_{\text{proj}}\rangle$ as our cost function in this example. The target energy was set to $\langle H_{\text{proj}}\rangle \leqslant 10^{-5}$, and other hyperparameters were $N_{\text{moves}} = 30$ and $N_{\text{samp}} = 10$. Because of the stochastic nature of the compilation process, we synthesised 100 circuits each in the full- and subspace, of which 97 of the subspace and 95 of the full space attempts converged. Figure 7 summarises the outcomes of the successful calculations.

The results distinctly show that for comparable accuracy (a), considering only a subspace results in vastly fewer gates (b), as well as many fewer iterations (c). Therefore we expect this method to yield much better results whenever the target unitary conserves some quantity, and the relevant subspace is known in advance. Prominently, this is the case for time evolution operators in quantum chemistry, where the number of electrons is a conserved and known quantity, and corresponds to the Hamming weight of the states. The sister paper to the present work [68] explores this application in greater detail.

4.3. Quantum Fourier transform

As shown in the previous subsection, when compiling random unitaries, the number of required parameters quickly makes compiling circuits with our available resources for more than a few qubits infeasible. However, practically relevant circuits usually have much more structure than random unitaries. We therefore also synthesise quantum Fourier transform (QFT) [69] circuits using various gate sets as examples of unitaries closer to real-world applications. Figure 8 shows the target circuit of a 4-qubit QFT as an example. We re-express it using the established gate set $\mathcal{L}_{\text{allrot}}$, as well as

$$\mathcal{L}_{\text{NNrot}} = \{R_k^\sigma, C_\ell[R_k^\sigma] \,|\, \sigma \in \{x,y,z\}, k,\ell \in [N] \text{ and } |k-\ell| = 1\} \tag{20}$$

which contains all local single-qubit rotations and local rotations controlled by nearest neighbours in an open linear chain topology. To test the ability of our algorithms to work with a much more restricted gate set, we furthermore synthesise QFT circuits using the set

$$\mathcal{L}_{\text{SWAP}} = \{R_k^\sigma, E_{k,\ell} \,|\, \sigma \in \{x,y,z\}, k,\ell \in [N] \text{ and } |k-\ell| = 1\} \tag{21}$$

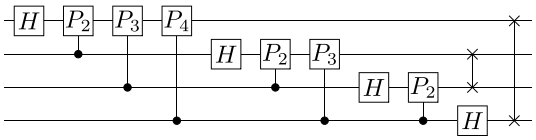

Figure 8. Example of a quantum Fourier transform circuit on four qubits as used in our calculations. The controlled operators $P_n = e^{-i\pi\,(\sigma^z - \mathbb{1})2^{-n}}$ are phase gates with rotation angles of $2\pi/2^n$.

which, in addition to local rotations, contains the parameterised SWAP gate introduced in equation (16) between neighbouring qubits as the only entangling operator. None of these gate sets contain Hadamard or controlled phase gates—which constitute the majority of the gates in the canonical circuit—making this a suitable benchmarking synthesis task.

For demonstration purposes, we tried synthesising QFT circuits for 3 to 6 qubits, all mentioned gate sets, and all discussed circuit structure generation algorithms, 100 times each. The target energy for the used cost function was $\langle H_{\text{sum}} \rangle \leqslant 10^{-8}$. The results are shown in figure 9. In addition to the outcomes shown in figure 9, we furthermore performed ten calculations each for $n_{\text{qb}} = 7 \ldots 10$ qubits using tabu search and the $\mathcal{L}_{\text{allrot}}$ gate set. These resulted in the convergence of 8, 7, 5, and 1 calculations, respectively, and minimum gate counts of 72, 93, 114, and 138.

We found that the *hill climbing* algorithm can perform well for some problems, especially when the available gate set has high expressibility as is the case for $\mathcal{L}_{\text{allrot}}$ and $\mathcal{L}_{\text{NNrot}}$. In these cases, hill climbing found circuits with gate counts close to the lower bound of all algorithms we investigated, albeit at a higher computational cost than tabu and random search. However, more restricted gate sets like $\mathcal{L}_{\text{SWAP}}$ severely hamper its ability to find solutions. Being the only deterministic algorithm we presented, circuits it fails to synthesise cannot be helped by re-running the procedure.

The probabilistic schemes of tabu search and random search, on the other hand, produce different outcomes for every run. We found that using a fully connected gate set consistently yields a higher probability for convergence than restricting the interactions to neighbouring qubits in a linear chain, despite using an initial circuit of SWAPs to counteract connectivity constraints. Calculations starting from an empty circuit (not plotted) show even lower convergence rates. Further decreasing the expressibility of the available gates by using $\mathcal{L}_{\text{SWAP}}$ as the gate set sees another significant drop in the relative number of converged calculations for $n_{\text{qb}} \geqslant 4$, indicating that all of our algorithms struggle to find solutions when they are greatly restricted in the choice of gates they can add.

Comparing our approaches of *tabu* and *random search*, we find—as for random unitaries—no significant difference in the number of gates in the resulting circuit or the convergence probability, despite tabu search trying to remember and avoid unsuccessful circuit modifications. It is to be expected for this mechanism not to have a noticeable impact when using libraries containing many gates, like $\mathcal{L}_{\text{allrot}}$, because the neighbourhood—i.e. the circuits which can be produced by adding a single gate from the library at any position—is much larger than the number of additions the algorithm can reasonably try during the iteration. For example, on 5 qubits with an existing circuit containing 20 gates, the $\mathcal{L}_{\text{allrot}}$ gate set produces several hundred potential moves, of which our algorithms typically explore ~ 20. However, even when using the $\mathcal{L}_{\text{SWAP}}$ library with only very few gates in it, which thus produces a smaller neighbourhood for each circuit, we found no significant difference between the two approaches. This observation is independent of how many iterations the *tabu* moves are remembered for.

4.4. *n*-qubit Toffoli

A potentially difficult operator to synthesise from only two-qubit gates is the n-qubit Toffoli gate, i.e. a Pauli-X gate with $n-1$ controls. The difficulty lies in the fact that it only acts on a very small subspace of $\mathcal{H}$, which is not straightforward to exclusively address using gates acting on much larger spaces of $\mathcal{H}$. Additionally, while it is relatively straightforward to detect whether a given circuit can act exclusively in the given subspace, finding a measure indicating that a given circuit is *close* to having this property proves difficult. If this kind of measure were found, it could guide the compilation process in the right direction. Unfortunately, our used cost functions do not contain such information.

In our implementation, because a large portion of all possible input states must remain unchanged by the circuit, adding any small number of gates will likely result in their parameters being tuned to zero during optimisation, as that matches the correct action on most input states. The algorithm then subsequently removes the gates with vanishing parameters, leading to no progress being made.

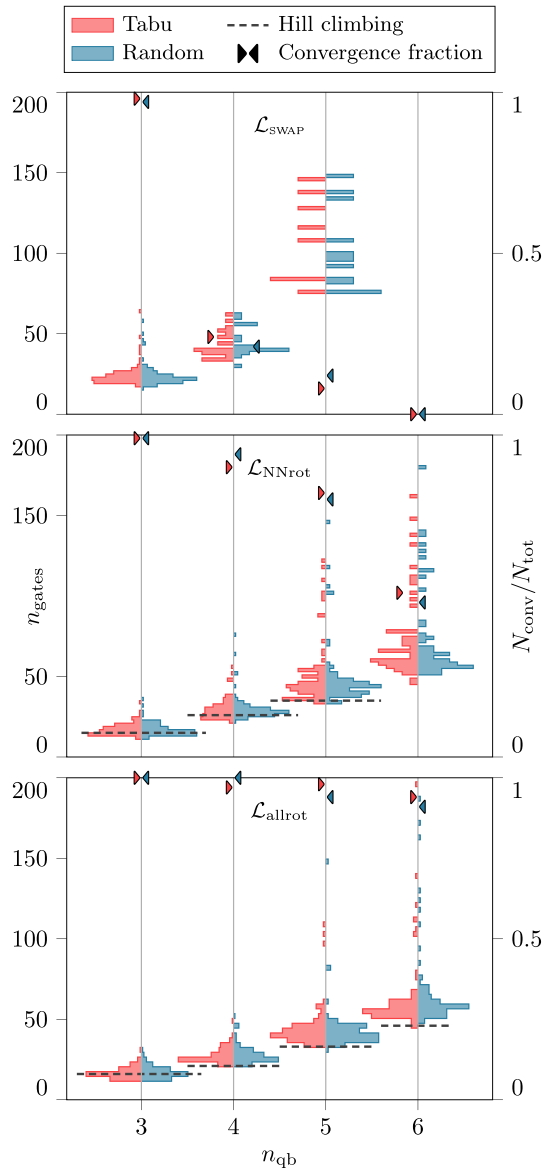

Figure 9. Number of gates and fraction of converged calculations for synthesis of a QFT circuit for $n_{\text{qb}} = 3 \ldots 6$ qubits and different gate sets $\mathcal{L}_{\text{SWAP}}$, $\mathcal{L}_{\text{NNrot}}$, and $\mathcal{L}_{\text{allrot}}$. Shaded graphs are histograms of the number of gates in the resultant circuit, left, red for tabu search, right, blue for random search, centred around their respective number of qubits. Each calculation was run 100 times with identical parameters as listed in the text and appendix. Red and blue carets indicate the fraction of converged calculations for each instance according to the right scale. Separation from the central line is for visual purposes only. Grey dashed lines, if present, indicate that the hill climbing algorithm converged for a particular instance and shows the number of qubits in the final circuit it produced.

Figure 10 shows the results for our synthesis calculations to generate n-qubit Toffoli gates for $n = 3, 4, 5$ using the $\mathcal{L}_{\text{allrot}}$ gate library, each 100 times. For the smaller cases of $n = 3$ and 4 our algorithm can find correct circuits reliably. In these cases the number of simultaneously added gates is large enough to overcome the previously discussed limitation. This is helped by the fact that the number of states on which the Toffoli acts like the identity operator is not overwhelmingly bigger than the number of states on which it acts nontrivially. Therefore, in these cases, we succeed in synthesising an appropriate circuit virtually all of the time. For $n \geqslant 5$, on the other hand, our algorithm was not able to find any solutions at all.

As an attempt to inject some prior knowledge into the algorithm, we also performed 5-qubit Toffoli gate synthesis starting the iteration from one of the successful 4-qubit Toffoli synthesis results. With this 'warm start' technique, 80 out of 100 calculations converged. While such an assisted start deviates from the strict *ab initio* framework—even more so than the SWAP network used in some of our calculations—and requires a specific incremental structure of the target circuit, it can still be a very useful resource for some tasks.

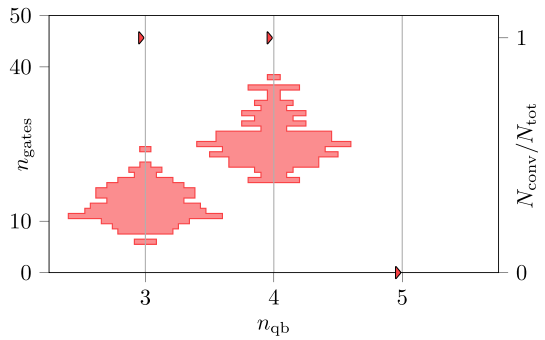

Figure 10. Histograms of the numbers of gates required and convergence fractions when synthesising an n_{qb}-qubit Toffoli using the $\mathcal{L}_{\text{allrot}}$ gate set and tabu search. Histograms are centred around the corresponding number of qubits, red carets indicate how many of the started calculations converged. Histogram data for 5 qubits is missing because none of the calculations succeeded.

5. Discussion and outlook

In this work we have combined a variant of the Hilbert–Schmidt test [45] with artificial Hamiltonians H_{sum} and H_{proj} similar to [49] to construct cost functions representing the closeness of a unitary to a dynamically created circuit, where the cost can in principle be evaluated on a quantum computer. However, we only employed emulators of such quantum hardware, and have thus circumvented some practical challenges like shot noise and barren plateaus, whose impact on the performance on the presented scheme remains to be investigated. We leave this question of how noisy hardware would affect the results as a potential future research direction. We presented three different algorithms which use this cost function to dynamically construct quantum circuits replicating the action of a given unitary from the ground up and demonstrated their performance on various synthesis tasks.

Our results suggest that the presented algorithms are able to generate circuit representations for dense random unitaries with gate counts close to what we would expect to be optimal, based on the degrees of freedom in such a unitary. For block-diagonal random unitaries we were furthermore able to show that generating a circuit whose closeness to the target is only judged within a restricted subspace greatly reduces both the synthesis resource requirements and the gate count in the resulting circuit. This can be important when time evolving Hamiltonians with such a block-diagonal structure, as is usually the case in quantum chemistry.

None of the cases we numerically investigated showed a significant difference between *random search* and *tabu search*. This strongly suggests that our simple attempt at guiding the search through the circuit structure space more efficiently than random moves is not sophisticated enough to yield any practical advantage. We note that the implementation of tabu search used here, only incorporates its most basic aspect of short-term memory. Its performance could potentially be improved by including more elaborate concepts such as intermediate-term and long-term memory [66].

The presented results for synthesising QFT circuits using various gate sets show that for small numbers of qubits and highly expressive sets of gates, the hill climbing method can consistently produce circuits with very few gates, almost always close to the minimum number we found for any method. However, if the method gets stuck in a local minimum or proceeds too slowly due to the increasing size of the neighbourhood, there are no provisions in our presented framework to overcome these problems. Tabu and random search produced accurate results even for a very restricted gate set on a small number of qubits, but scaled unfavourably when increasing the number of qubits in these cases.

Finally, the attempts at synthesising an n-qubit Toffoli gate clearly showed the limitations of our method. Due to the properties of the cost functions, as discussed in section 4.4, we were only able to generate circuits for Toffoli gates on 5 qubits by assisting the algorithm with previously generated knowledge.

We emphasise that there is significant value in the ability to synthesise even small unitaries, since such compiled functions can be used as components of larger algorithms. For example, in grid-based chemistry, although the total number of computational qubits may be in the many thousands, the QFT used to move between real-space and momentum-space is local to each particle's representation, often acting on only around 20 qubits [70]. A different example is the method in [28], where large circuits are first decomposed into blocks small enough for individual recompilation. The presented method may be used as a subroutine in such a scheme to synthesise efficient circuits for each of these blocks. Indeed, even the ability to compile a

multi-qubit gate involving 3 or 4 qubits into a compact set of 1- and 2-qubit gates can be valuable. Therefore the significance of the techniques described in this paper does not depend on their ability to scale directly to circuit sizes that might be considered 'post-classical' ($\gtrsim$50 qubits). Nevertheless it is of course interesting to reflect on the prospects for such large scale circuit synthesis and (re-)compilations. Our results suggest that the methods will require significant further development for any such task to be realistic. We now remark on a few such possibilities.

Firstly, note that the algorithms used for circuit structure modifications are largely independent from the parameter optimisation routine, except for reusing information in the QMT. Therefore, if a different method for optimising the parameters proves more suitable, it can be straightforwardly substituted for the imaginary time evolution used in the present paper. One promising candidate is introduced in [71] named CoVaR, where an eigenstate of the system is prepared by a root-finding algorithm similar to Newton's method. By tweaking the spectrum of our synthesis Hamiltonians such that only product states are eigenstates, this method could be used to find the appropriate parameters in each iteration.

Secondly, the gate sets need not consist only of unitary operators. Instead, it would be possible to also include ancilla qubits on which intermediate measurements may be performed, whose outcomes can become part of the cost function. In this case, imaginary time evolution must be adapted to find the correct descent direction [60].

Thirdly, there are also possible enhancements to the Hamiltonians used to generate our cost functions. As briefly mentioned in section 2.2, instead of the relatively straightforward H_{proj} and H_{sum}, other properties of the solution may be included to judge how suitable a particular outcome is, such as the desired entanglement via a witness, or the conservation of symmetries in the problem.

Fourthly and perhaps most challengingly, an intriguing direction of research is to explore methods for introducing circuit variants that are more nuanced than the Darwinian 'random variation, non-random selection' employed in this paper. While enhancements to the tabu search introduced here may form part of the solution, another way forward might be to explore techniques that use information from the output state to deduce which modifications to the circuit are most likely to result in a reduction of the cost function.

Finally we remark that the presented methods to construct circuits *ab initio* can be used not only to express a desired unitary using various target gate sets, but also as a variational quantum eigensolver to prepare the ground state of some physical Hamiltonian. For this task, the evaluation of the cost function is straightforwardly replaced by the Hamiltonian of interest. This idea, among other applications, is explored in the sister paper to the present work [68].

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/rrmeister/circuit-synthesis-results.

Acknowledgments

The authors would like to thank Tyson Jones and Bálint Koczor for useful discussions and feedback on the manuscript. C G and S C B acknowledge financial support from EPSRC Hub grants under Agreement No. EP/T001062/1, from the IARPA funded LogiQ project, and the EU flagship AQTION project.

The authors acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work (https://doi.org/10.5281/zenodo.22558).

Appendix A. Algorithmic details

A.1. Global hyperparameters

In order to make the algorithms easier to read, we globally define some hyperparameters in table A1. They stay constant during the whole synthesis process and can be used to tweak some properties of the algorithms.

A.2. Pseudocode

To minimise clutter in the main text, we collect pseudocode for most of our described routines in this appendix for the interested reader. In contrast to the main text, where we write the cost function as $\langle \tilde{H} \rangle$, in pseudocode to explicitly denote which circuit is applied, we use the notation

$$\mathcal{E}(\mathcal{C}(\underline{\theta})) = \langle \psi_1 | \tilde{H} | \psi_1 \rangle \tag{A1}$$

Table A1. Hyperparameters used in the pseudocode of our subroutines, collected here for more concise descriptions of the algorithms.

Param	Default	Use
$\tilde{H}$	H_{sum}	The synthesis Hamiltonian in the augmented space. In our case either H_{sum} or H_{proj}.
δ_{abs}	10^{-5}	Threshold for the absolute energy change per iteration regarded as constant during the parameter optimisation.
δ_{rel}	10^{-3}	Limit for the relative energy change per iteration considered constant during the parameter optimisation.
$k_{\text{max,opt}}$	500	Maximum number of iterations for parameter optimisation.
n_{conv}	5	Number of times the convergence criterion must be fulfilled in parameter optimisation to be considered converged.
κ	1.4	Factor by which the step size λ of the parameter optimisation is in- or decreased while searching along the gradient direction.
λ_0	0.05	Initial step size in the parameter optimisation routine.
k_{max}	10 000	Maximum number of iterations for circuit modifications.
E_{conv}	10^{-8}	Energy at which the calculation is considered converged.
N_{moves}	30	Number of gates added in a single circuit modification iteration for random and tabu search.
N_{samp}	10	Number of times a circuit is modified in a single iteration until the best result is picked for the next iteration in random search.
$\mathcal{L}$	$\mathcal{L}_{\text{allrot}}$	The library to draw new gates from.
ε_{QMT}	10^{-3}	Small quantity to detect linearly dependent rows in the QMT.
$\varepsilon_{\text{param}}$	$\varepsilon_{\text{remove}}$	Threshold for the magnitude of parameters below which the corresponding gate is removed.
$\varepsilon_{\text{remove}}$	$\frac{E_k - E_{k-1}}{50}$	Energy increase considered acceptable for the removal of unnecessary gates.

with

$$|\psi_1\rangle = P^\dagger \mathcal{C}^\dagger(\underline{\theta}) U P(|0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}'}) \tag{A2}$$

where operator P prepares Bell pairs as depicted on the left side of figure 2.

Algorithm 1. The routine we use to optimise the parameters $\underline{\theta}$ for a given circuit structure $\mathcal{C}$.

function OPTIMISEPARAMETERS($\mathcal{C}$, $\underline{\theta}$)
  $E_0 \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta}))$, $k_{\text{conv}} \leftarrow 0$
  for $k \leftarrow 1 \,..\, k_{\text{max,opt}}$ **do**
    $A_{ij} \leftarrow \text{Re}(\langle \partial_i \psi | \partial_j \psi \rangle - \langle \partial_i \psi | \psi \rangle \langle \psi | \partial_j \psi \rangle) \; \forall \, i,j$
    $B_i \leftarrow -\langle \partial_i \psi | \hat{H} | \psi \rangle \; \forall \, i$
    $\underline{\Delta} \leftarrow$REGULARISE$(\mathbf{A})^{-1} \, \underline{B}$
    $\lambda \leftarrow$CHOOSELAMBDA$(\lambda_0, \mathcal{C}, \underline{\theta}, \underline{\Delta})$
    $\underline{\theta} \leftarrow \underline{\theta} - \lambda \underline{\Delta}$
    $E_k \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta}))$
    if $E_k - E_{k-1} < \delta_{\text{abs}}$ **or** $\frac{E_k - E_{k-1}}{E_k} < \delta_{\text{rel}}$ **then**
      $k_{\text{conv}} \leftarrow k_{\text{conv}} + 1$
    else
      $k_{\text{conv}} \leftarrow 0$
    if $k_{\text{conv}} = n_{\text{conv}}$ **then return** $\underline{\theta}$
    return fail ▷ No convergence at max iterations
function CHOOSELAMBDA(λ, $\mathcal{C}$, $\underline{\theta}$, $\underline{\Delta}$)
  $\lambda^- \leftarrow \lambda/\kappa$, $\lambda^+ \leftarrow \lambda \cdot \kappa$, $E \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta} - \lambda \underline{\Delta}))$
  $E^+ \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta} - \lambda^+ \underline{\Delta}))$, $E^- \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta} - \lambda^- \underline{\Delta}))$
  if $E^- > E < E^+$ **or** $\lambda \leqslant \lambda_{\text{min}}$ **then**
    return λ ▷ No improvement either side
  if $E^+ < E^-$ **then**
    return CHOOSELAMBDA$(\lambda^+, \mathcal{C}, \theta, \underline{\Delta})$ ▷ Grow λ
  return CHOOSELAMBDA$(\lambda^-, \mathcal{C}, \theta, \underline{\Delta})$ ▷ Shrink λ

Algorithm 2. Routine GenerateMoves to generate all moves applicable to a circuit structure $\mathcal{C}$ not leading to obvious redundancies in the resulting circuit. The helper function ApplyMove applies the move M to the circuit structure $\mathcal{C}$ and parameter vector $\underline{\theta}$, and makes other routines easier to read.

function GenerateMoves($\mathcal{C}$)
  $\mathcal{M} \leftarrow \{\}$
  for $k \leftarrow 1..N_{\text{qubits}}$ **do**
    ▷ Indices of gates touching qubit k
    $\tilde{N} \leftarrow (p \mid \mathcal{C}_p \text{ acts on qubit } k)$
    $G_{\text{left}} \leftarrow \mathbb{1}$
    for all $m \in \tilde{N}$ **do**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{(G, m) \mid G \in \mathcal{L} \text{ and } G \neq G_{\text{left}} \text{ and } G \neq \mathcal{C}_m\}$
      $G_{\text{left}} \leftarrow \mathcal{C}_m$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(G, \max(\tilde{N}) + 1) \mid G \in \mathcal{L} \text{ and } G \neq G_{\text{left}}\}$
  return $\mathcal{M}$

function ApplyMove($\mathcal{C}, \underline{\theta}, M$)
  $\mathcal{C} \leftarrow (\mathcal{C}_0, \ldots, \mathcal{C}_{M_n-1}, M_G, \mathcal{C}_{M_n}, \ldots)$
  $\underline{\theta} \leftarrow (\theta_0, \ldots \theta_{M_n-1}, 0, \theta_{M_n}, \ldots)$
  return $\mathcal{C}, \underline{\theta}$

Algorithm 3. A simplified version of the random search algorithm used to generate our results. It calls several subroutines from algorithms 1, 2 and 4.

function RandomSearch($\mathcal{C}^{(0)}, \underline{\theta}^{(0)}$)
  $E_0 \leftarrow \mathcal{E}(\mathcal{C}^{(0)}(\underline{\theta}^{(0)}))$
  for $k \leftarrow 1..k_{\max}$ **do**
    ▷ $\mathcal{R}$ and $\tilde{\mathcal{R}}$ contain indices of newly added gates
    $E_k \leftarrow E_{k-1}, \quad \mathcal{R} \leftarrow \{\}$
    $\mathcal{C}^{(k)} \leftarrow \mathcal{C}^{(k-1)}, \quad \underline{\theta}^{(k)} \leftarrow \underline{\theta}^{(k-1)}$
    for $m \leftarrow 1..N_{\text{samp}}$ **do**
      $\tilde{\mathcal{C}} \leftarrow \mathcal{C}^{(k-1)}, \quad \underline{\tilde{\theta}} \leftarrow \underline{\theta}^{(k-1)}, \quad \tilde{\mathcal{R}} \leftarrow \{\}$
      for $n \leftarrow 1..N_{\text{moves}}$ **do**
        $M \leftarrow$**random element of** GenerateMoves($\tilde{\mathcal{C}}$)
        $\tilde{\mathcal{C}}, \underline{\tilde{\theta}} \leftarrow$ ApplyMove($\tilde{\mathcal{C}}, \underline{\tilde{\theta}}, M$)
        $\tilde{\mathcal{R}} \leftarrow \{j \mid j \in \tilde{\mathcal{R}} \text{ and } j < M_n\} \cup \{M_n\} \cup \{j+1 \mid j \in \tilde{\mathcal{R}} \text{ and } j \geqslant M_n\}$
      $\underline{\tilde{\theta}} \leftarrow$ OptimiseParameters($\tilde{\mathcal{C}}, \underline{\tilde{\theta}}$)
      if $\mathcal{E}(\tilde{\mathcal{C}}(\underline{\tilde{\theta}})) < E_k$ **then**
        $\mathcal{C}^{(k)} \leftarrow \tilde{\mathcal{C}}, \quad \underline{\theta}^{(k)} \leftarrow \underline{\tilde{\theta}}$
        $E_k \leftarrow \mathcal{E}(\tilde{\mathcal{C}}(\underline{\tilde{\theta}})), \quad \mathcal{R} \leftarrow \tilde{\mathcal{R}}$
    $\mathcal{C}^{(k)}, \underline{\theta}^{(k)} \leftarrow$ Prune($\mathcal{C}^{(k)}, \underline{\theta}^{(k)}, \mathcal{R}$)
    $E_k \leftarrow \mathcal{E}(\mathcal{C}^{(k)}(\underline{\theta}^{(k)}))$
    if $E_k < E_{\text{conv}}$ **then**
      return Prune($\mathcal{C}^{(k)}, \underline{\theta}^{(k)}, \{0, \ldots, N_{\text{gates}}(\mathcal{C}^{(k)})\}$)
  return fail ▷ No convergence at iteration limit

Algorithm 4. Our routine to detect and remove unnecessary gates from a circuit. $\mathcal{C}$ and $\underline{\theta}$ are the circuit structure and current parameters, respectively, and $\mathcal{R}$ is a set of indices of gates that should be considered for deletion. The helper function DELETE removes the gates at the indices specified in $\mathcal{D}$ from the circuit and updates the indices in $\mathcal{R}$ accordingly.

function Prune($\mathcal{C}$, $\underline{\theta}$, $\mathcal{R}$)
  ▷ Remove vanishing parameters
  $\mathcal{D} \leftarrow \{k \,|\, |\theta_k \bmod 2\pi| < \varepsilon_{\text{param}}\}$
  $\mathcal{C}, \underline{\theta}, \mathcal{R} \leftarrow$ DELETE($\mathcal{C}, \underline{\theta}, \mathcal{R}, \mathcal{D}$)
  ▷ Quantum metric tensor assisted removal
  $\mathcal{D} \leftarrow \{\}, \quad \mathcal{K} \leftarrow \{\}, \quad \underline{\theta}^+ \leftarrow \underline{0}$
  $A_{ij} \leftarrow \text{Re}(\langle \partial_i \psi | \partial_j \psi \rangle - \langle \partial_i \psi | \psi \langle \langle \psi | \partial_j \psi \rangle) \; \forall i, j$
  for $k \in \mathcal{R}$ **do**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{n \,|\, n > k \text{ and } \left|(A^{\intercal}_{k,\cdot} \cdot A_{n,\cdot}) - \|A_{k,\cdot}\| \, \|A_{n,\cdot}\|\right| < \varepsilon_{\text{QMT}}\}$
    for all $n \in \mathcal{D}$ **do**
      $\mathcal{C}', \underline{\theta}' \leftarrow$ DELETE($\mathcal{C}, \underline{\theta}, \{\}, \{n\}$)
      $\theta'_k \leftarrow \theta_k + \theta_n$
      if $\mathcal{E}(\mathcal{C}'(\underline{\theta}')) > \mathcal{E}(\mathcal{C}(\theta)) + \varepsilon_{\text{remove}}$ **then**
        $\mathcal{K} \leftarrow \mathcal{K} \cup \{n\}$
      else
        $\theta^+_k \leftarrow \theta^+_k + \theta_n$
  $\underline{\theta} \leftarrow \underline{\theta} + \underline{\theta}^+$
  $\mathcal{C}, \underline{\theta}, \mathcal{R} \leftarrow$ DELETE($\mathcal{C}, \underline{\theta}, \mathcal{R}, \mathcal{D} \backslash \mathcal{K}$)
  ▷ Trial and error removal
  while $\mathcal{R} \neq \{\}$ **do**
    $k \leftarrow \max(\mathcal{R})$
    $\mathcal{R} \leftarrow \mathcal{R} \backslash \{k\}$
    $\mathcal{C}' \leftarrow (\mathcal{C}_0, \ldots \mathcal{C}_{k-1}, \mathcal{C}_{k+1}, \ldots)$
    $\underline{\theta}' \leftarrow (\theta_0, \ldots \theta_{k-1}, \theta_{k+1}, \ldots)$
    $\underline{\theta}' \leftarrow$ OPTIMISEPARAMETERS($\mathcal{C}', \underline{\theta}'$)
    if $\mathcal{E}(\mathcal{C}'(\underline{\theta}')) < \mathcal{E}(\mathcal{C}(\underline{\theta})) + \varepsilon_{\text{remove}}$ **then**
      $\mathcal{C} \leftarrow \mathcal{C}'$
      $\underline{\theta} \leftarrow \underline{\theta}'$
  return $\mathcal{C}, \underline{\theta}$

function DELETE($\mathcal{C}$, $\underline{\theta}$, $\mathcal{R}$, $\mathcal{D}$)
  for $d \leftarrow$ INVERSESORTED($\mathcal{D}$) **do**
    $\mathcal{C} \leftarrow (\mathcal{C}_0, \ldots \mathcal{C}_{d-1}, \mathcal{C}_{d+1}, \ldots)$
    $\underline{\theta} \leftarrow (\theta_0, \ldots \theta_{d-1}, \theta_{d+1}, \ldots)$
    $\mathcal{R} \leftarrow \{n \,|\, n \in \mathcal{R} \text{ and } n < d\} \cup \{n - 1 \,|\, n \in \mathcal{R} \text{ and } n > d\}$
  return $\mathcal{C}, \underline{\theta}, \mathcal{R}$

Algorithm 5. A simple variant of hill climbing for circuit synthesis.

function HILLCLIMB($\mathcal{C}^{(0)}$, $\underline{\theta}^{(0)}$)
  $E_0 \leftarrow \mathcal{E}(\mathcal{C}^{(0)}(\underline{\theta}^{(0)}))$
  for $k \leftarrow 1 \,..\, k_{\max}$ **do**
    $E_k \leftarrow E_{k-1}$
    for all $M \in$ GENERATEMOVES($\mathcal{C}^{(k-1)}$) **do**
      $\tilde{\mathcal{C}}, \tilde{\underline{\theta}} \leftarrow$ APPLYMOVE($\mathcal{C}^{(k-1)}, \underline{\theta}^{(k-1)}, M$)
      $\tilde{\underline{\theta}} \leftarrow$ OPTIMISEPARAMETERS($\tilde{\mathcal{C}}, \tilde{\underline{\theta}}$)
      if $\mathcal{E}(\tilde{\mathcal{C}}(\tilde{\underline{\theta}})) < E_k$ **then**
        $\mathcal{C}^{(k)} \leftarrow \tilde{\mathcal{C}}, \quad \underline{\theta}^{(k)} \leftarrow \tilde{\underline{\theta}}, \quad E_k \leftarrow \mathcal{E}(\tilde{\mathcal{C}}(\tilde{\underline{\theta}}))$
    if $E_k < E_{\text{conv}}$ **then**
      return PRUNE($\mathcal{C}^{(k)}, \underline{\theta}^{(k)}, \{0, \ldots, N_{\text{gates}}(\mathcal{C}^{(k)})\}$)
    if $E_k = E_{k-1}$ **then**
      return fail ▷ No more improvement
return fail ▷ No convergence at iteration limit

Table B2. Details of results with the least number of two-qubit gates for the QFT and Toffoli circuits discussed in the main text. The number of two-qubit gates (N_{2qb}) is the lowest found in any calculation for the given circuit and gate set, N_{gates} is the corresponding number of total gates in those circuits. The mean compile time $\overline{T}$ is averaged over all calculations with the specified parameters, including ones that terminated due to a set time limit.

Circuit	n_{qb}	Gate set	N_{2qb}	N_{gates}	$\overline{T}$ [s]
QFT	3	allrot	6	14	103
QFT	3	NNrot	6	12	174
QFT	3	SWAP	6	20	829
QFT	4	allrot	12	22	342
QFT	4	NNrot	12	24	1459
QFT	4	SWAP	13	132	6228
QFT	5	allrot	20	37	9966
QFT	5	NNrot	20	35	39 709
QFT	5	SWAP	30	128	116 782
QFT	6	allrot	29	46	53 235
QFT	6	NNrot	30	77	183 580
QFT	6	SWAP	92	123	239 180
QFT	7	allrot	44	72	126 094
QFT	7	NNrot	46	209	248 486
QFT	8	allrot	60	418	263 227
QFT	8	NNrot	77	115	345 283
QFT	9	allrot	82	259	317 498
QFT	10	allrot	162	429	431 999
Toffoli	3	allrot	5	6	26
Toffoli	4	allrot	13	20	1151
Toffoli	5	allrot	33	167	96 868

Appendix B. Gate counts

In this appendix we collect the results exhibiting the smallest number of two-qubit gates for the QFT and Toffoli syntheses across all *random search* and *tabu search* calculations. The used gate sets are those given in the main text, with added parametrised SWAP ($E_{i,j}$) gates for the substrate. Table B2 contains the full result data.

The mean compile time $\overline{T}$ is reported for execution on 8 cores ($n_{qb} \leqslant 6$) or 48 cores ($n_{qb} > 6$) of Intel Platinum 8628 CPUs within the University of Oxford Advanced Research Computing facility.

ORCID iDs

Richard Meister https://orcid.org/0000-0002-1998-7867
Cica Gustiani https://orcid.org/0000-0003-0558-4685
Simon C Benjamin https://orcid.org/0000-0002-7766-5348

References

[1] Toffoli T 1981 Bicontinuous extensions of invertible combinatorial functions *Math. Syst. Theory* **14** 13–23
[2] Fredkin E and Toffoli T 1982 Conservative logic *Int. J. Theor. Phys.* **21** 219–53
[3] Feynman R P 1986 Quantum mechanical computers *Found. Phys.* **16** 507–31
[4] Deutsch D E 1989 Quantum computational networks *Proc. R. Soc. A* **425** 73–90
[5] Barenco A, Bennett C H, Cleve R, DiVincenzo D P, Margolus N, Shor P, Sleator T, Smolin J A and Weinfurter H 1995 Elementary gates for quantum computation *Phys. Rev.* A **52** 3457–67
[6] Akerman N, Navon N, Kotler S, Glickman Y and Ozeri R 2015 Universal gate-set for trapped-ion qubits using a narrow linewidth diode laser *New J. Phys.* **17** 113060
[7] Shapira Y, Shaniv R, Manovitz T, Akerman N and Ozeri R 2018 Robust entanglement gates for trapped-ion qubits *Phys. Rev. Lett.* **121** 180502
[8] Webb A E, Webster S C, Collingbourne S, Bretaud D, Lawrence A M, Weidt S, Mintert F and Hensinger W K 2018 Resilient entangling gates for trapped ions *Phys. Rev. Lett.* **121** 180501
[9] Manovitz T, Shapira Y, Gazit L, Akerman N and Ozeri R 2021 A trapped ion quantum computer with robust entangling gates and quantum coherent feedback (arXiv:2111.04155)
[10] Ma S, Burgers A P, Liu G, Wilson J, Zhang B and Thompson J D 2021 universal gate operations on nuclear spin qubits in an optical tweezer array of ^{171}Yb atoms (arXiv:2112.06799)
[11] Chow J M *et al* 2012 Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits *Phys. Rev. Lett.* **109** 060501
[12] Zhu D, Jaako T, He Q and Rabl P 2021 Quantum computing with superconducting circuits in the picosecond regime *Phys. Rev. Appl.* **16** 014024
[13] Long J et al 2021 A universal quantum gate set for transmon qubits with strong ZZ interactions (arXiv:2103.12305)

[14] Reuer K, Besse J C, Wernli L, Magnard P, Kurpiers P, Norris G J, Wallraff A and Eichler C 2021 Realization of a universal quantum gate set for itinerant microwave photons (arXiv:2106.03481)
[15] Wu T and Guo J 2018 Computational assessment of silicon quantum gate based on detuning mechanism for quantum computing *IEEE Trans. Electron Devices* **65** 5530–6
[16] Ferraro E, Rei D, Paris M and Michielis M D 2022 Universal set of quantum gates for the flip-flop qubit in the presence of 1/f noise *EPJ Quantum Technol.* **9** 2
[17] Evans T *et al* 2022 Fast Bayesian tomography of a two-qubit gate set in silicon *Phys. Rev. Appl.* **17** 024068
[18] Noiri A, Takeda K, Nakajima T, Kobayashi T, Sammak A, Scappucci G and Tarucha S 2022 Fast universal quantum gate above the fault-tolerance threshold in silicon *Nature* **601** 338–42
[19] Mills A R, Guinn C R, Gullans M J, Sigillito A J, Feldman M M, Nielsen E and Petta J R 2021 Two-qubit silicon quantum processor with operation fidelity exceeding 99% (arXiv:2111.11937)
[20] Jordan S 2021 Quantum algorithm zoo (available at: https://quantumalgorithmzoo.org)
[21] Tucci R R A rudimentary quantum compiler (arXiv:quant-ph/9805015)
[22] Iten R, Colbeck R, Kukuljan I, Home J and Christandl M 2016 Quantum circuits for isometries *Phys. Rev.* A **93** 032318
[23] Iten R, Reardon-Smith O, Malvetti E, Mondada L, Pauvert G, Redmond E, Kohli R S and Colbeck R 2019 Introduction to universalQCompiler (arXiv:1904.01072)
[24] Krol A M, Sarkar A, Ashraf I, Al-Ars Z and Bertels K 2022 Efficient decomposition of unitary matrices in quantum circuit compilers *Appl. Sci.* **12** 759
[25] Gustiani C and DiVincenzo D P 2021 Blind three-qubit exact Grover search on a nitrogen-vacancy-center platform *Phys. Rev.* A **104** 062422
[26] Davis M G, Smith E, Tudor A, Sen K, Siddiqi I and Iancu C 2020 Towards optimal topology aware quantum circuit synthesis *2020 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) (https://doi.org/10.1109/QCE49297.2020.00036)
[27] Smith E, Davis M G, Larson J, Younis E, Oftelie L B, Lavrijsen W and Iancu C 2023 LEAP: scaling numerical optimization based synthesis using an incremental approach *ACM Trans. Quantum Comput.* **4** 1–23
[28] Younis E, Sen K, Yelick K and Iancu C 2021 QFAST: conflating search and numerical optimization for scalable quantum circuit synthesis *2021 IEEE Int. Conf. on Quantum Computing and Engineering* (*QCE*) (IEEE) (https://doi.org/10.1109/QCE52317.2021.00041)
[29] Patel T, Younis E, Iancu C, de Jong W and Tiwari D 2022 QUEST: systematically approximating Quantum circuits for higher output fidelity *Proc. 27th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems* (ACM) (https://doi.org/10.1145/3503222.3507739)
[30] Zhou X, Li S and Feng Y 2020 Quantum circuit transformation based on simulated annealing and heuristic search *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **39** 4683–94
[31] Jiang H, Deng Y and Xu M 2021 Quantum circuit transformation based on tabu search (arXiv:2104.05214)
[32] Paler A, Sasu L M, Florea A and Andonie R 2020 Machine learning optimization of quantum circuit layouts (arXiv:2007.14608)
[33] Zhou X, Feng Y and Li S 2021 Supervised learning enhanced quantum circuit transformation (arXiv:2110.03057)
[34] Childs A M, Schoute E and Unsal C M 2019 Circuit transformations for quantum architectures *14th Conf. on the Theory of Quantum Computation, Communication and Cryptography* (*TQC 2019*) (*Leibniz Int. Proc. in Informatics* (*LIPIcs*) Vol 135, ed W van Dam and L Mancinska (Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik) p 3:1–3:24
[35] Li G, Ding Y and Xie Y 2019 Tackling the qubit mapping problem for NISQ-Era quantum devices *Proc. 24th Int. Conf. on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS '19*) (New York: Association for Computing Machinery) pp 1001–14
[36] Niu S, Suau A, Staffelbach G and Todri-Sanial A 2020 A hardware-aware heuristic for the qubit mapping problem in the NISQ Era *IEEE Trans. Quantum Eng.* **1** 1–14
[37] Zhou X, Feng Y and Li S 2022 Quantum circuit transformation: a Monte Carlo tree search framework *ACM Trans. Des. Autom. Electron. Syst.* **27** 1–27
[38] Devulapalli D, Schoute E, Bapat A, Childs A M and Gorshkov A V 2022 Quantum routing with teleportation (arXiv:2204.04185)
[39] Matteo O D and Mosca M 2016 Parallelizing quantum circuit synthesis *Quantum Sci. Technol.* **1** 015003
[40] Arufe L, González M A, Oddi A, Rasconi R and Varela R 2022 Quantum circuit compilation by genetic algorithm for quantum approximate optimization algorithm applied to MaxCut problem *Swarm Evol. Comput.* **69** 101030
[41] Venturelli D, Do M, Rieffel E and Frank J 2018 Compiling quantum circuits to realistic hardware architectures using temporal planners *Quantum Sci. Technol.* **3** 025004
[42] Moro L, Paris M G A, Restelli M and Prati E 2021 Quantum compiling by deep reinforcement learning *Commun. Phys.* **4** 178
[43] Cincio L, Subas Y, Sornborger A T and Coles P J 2018 Learning the quantum algorithm for state overlap *New J. Phys.* **20** 113022
[44] Bilkis M, Cerezo M, Verdon G, Coles P J, and Cincio L 2021 A semi-agnostic ansatz with variable structure for quantum machine learning (arXiv:2103.06712)
[45] Khatri S, LaRose R, Poremba A, Cincio L, Sornborger A T and Coles P J 2019 Quantum-assisted quantum compiling *Quantum* **3** 140
[46] Grimsley H R, Economou S E, Barnes E and Mayhall N J 2019 An adaptive variational algorithm for exact molecular simulations on a quantum computer *Nat. Commun.* **10** 3007
[47] Tang H L, Shkolnikov V, Barron G S, Grimsley H R, Mayhall N J, Barnes E and Economou S E 2021 Qubit-ADAPT-VQE: an adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor *PRX Quantum* **2** 020310
[48] Rattew A G, Hu S, Pistoia M, Chen R, Wood S and Domain-agnostic A 2019 Noise-resistant, hardware-efficient evolutionary variational quantum eigensolver (arXiv:1910.09694)
[49] Jones T and Benjamin S C 2022 Robust quantum compilation and circuit optimisation via energy minimisation *Quantum* **6** 628
[50] Cincio L, Rudinger K, Sarovar M and Coles P J 2021 Machine learning of noise-resilient quantum circuits *PRX Quantum* **2** 010324
[51] Caro M C, Huang H Y, Cerezo M, Sharma K, Sornborger A T, Cincio L, and Coles P J 2021 Generalization in quantum machine learning from few training data (arXiv:2111.05292)
[52] Caro M C, Huang H Y, Ezzell N, Gibbs J, Sornborger A T, Cincio L, Coles P J and Holmes Z 2022 Out-of-distribution generalization for learning quantum dynamics (arXiv:2204.10268)
[53] Gibbs J, Holmes Z, Caro M C, Ezzell N, Huang H Y, Cincio L, Sornborger A T and Coles P J 2022 Dynamical simulation via quantum machine learning with provable generalization (arXiv:2204.10269)
[54] Jamiołkowski A 1972 Linear transformations which preserve trace and positive semidefiniteness of operators *Rep. Math. Phys.* **3** 275–8

[55] Choi M D 1975 Completely positive linear maps on complex matrices *Linear Algebr. Appl.* **10** 285–90
[56] Cerezo M, Sone A, Volkoff T, Cincio L and Coles P J 2021 Cost function dependent barren plateaus in shallow parametrized quantum circuits *Nat. Commun.* **12** 1791
[57] Hamming R W 1950 Error detecting and error correcting codes *Bell Syst. Tech. J.* **29** 147–60
[58] McArdle S, Jones T, Endo S, Li Y, Benjamin S C and Yuan X 2019 Variational ansatz-based quantum simulation of imaginary time evolution *npj Quantum Inf.* **5** 75
[59] Yuan X, Endo S, Zhao Q, Li Y and Benjamin S C 2019 Theory of variational quantum simulation *Quantum* **3** 191
[60] Koczor B and Benjamin S C 2019 Quantum natural gradient generalised to non-unitary circuits (arXiv:1912.08660)
[61] Yamamoto N 2019 On the natural gradient for variational quantum eigensolver (arXiv:1909.05074)
[62] Stokes J, Izaac J, Killoran N and Carleo G 2020 Quantum natural gradient *Quantum* **4** 269
[63] Euler L 1913 *Institutiones Calculi Integralis* vol 1 (Birkhäuser)
[64] van Straaten B and Koczor B 2021 Measurement cost of metric-aware variational quantum algorithms *PRX Quantum* **2** 030324
[65] Skiena S S 2020 *The Algorithm Design Manual* 3rd edn (Berlin: Springer)
[66] Glover F W 1990 Tabu search: a tutorial *Interfaces* **20** 74–94
[67] Meister Richard and Gustiani Cica 2023 Result data for "Exploring ab initio machine synthesis of quantum circuits" *GitHub* (available at: https://github.com/rrmeister/circuit-synthesis-results)
[68] Gustiani C, Meister R and Benjamin S C 2023 Exploiting subspace constraints and *ab initio* variational methods for quantum chemistry *New J. Phys.* **25** 073019
[69] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[70] Chan H H S, Meister R, Jones T, Tew D P and Benjamin S C 2022 Grid-based methods for chemistry simulations on a quantum computer (arXiv:2202.05864)
[71] Boyd G and Koczor B 2022 Training variational quantum circuits with CoVaR: covariance root finding with classical shadows (arXiv:2204.08494)