Approaching the theoretical limit in quantum gate decomposition

Péter Rakyta[1,2] and Zoltán Zimborás[3,4]

[1]Department of Physics of Complex Systems, Eötvös Loránd University, Budapest, Hungary
[2]Wigner Research Center for Physics, 29–33 Konkoly–Thege Miklos Str., H- 1121 Budapest, Hungary
[3]Wigner Research Center for Physics, 29–33 Konkoly–Thege Miklos Str., H-1121 Budapest, Hungary
[4]BME-MTA Lendület Quantum Information Theory Research Group, Budapest, Hungary

arXiv:2109.06770v4 [quant-ph] 9 May 2022

In this work we propose a novel numerical approach to decompose general quantum programs in terms of single- and two-qubit quantum gates with a $CNOT$ gate count very close to the current theoretical lower bounds. In particular, it turns out that 15 and 63 $CNOT$ gates are sufficient to decompose a general 3- and 4-qubit unitary, respectively, with high numerical accuracy. Our approach is based on a sequential optimization of parameters related to the single-qubit rotation gates involved in a pre-designed quantum circuit used for the decomposition. In addition, the algorithm can be adopted to sparse inter-qubit connectivity architectures provided by current mid-scale quantum computers, needing only a few additional $CNOT$ gates to be implemented in the resulting quantum circuits.

1 Introduction

Quantum mechanical resources like superposition and entanglement are useful when solving computational problems that are intractable with classical resources, i.e., with classical computing devices. For example, Shor's integer factorization [1] and Grover's database search [2] algorithms promise a significant speedup compared to the best classical algorithms developed for these problems. Even if currently available quantum computers are not advanced enough yet to outperform classical devices in solving these problems, a lot of progress has been made in demonstrating these algorithms on small quantum devices [3, 4, 5, 6, 7]. Beside these well known examples, various other promising schemes were developed to exploit quantum resources in solving computational problems, for example, variational quantum optimization [8], variational quantum eigensolvers [9], and quantum simulations of molecular and many-body phenomena [10, 11, 12, 13, 14] were successfully implemented on hardware, and at least qualitatively justified results were obtained.

In general, quantum algorithms can be described by unitary transformations and projective measurements acting on the 2^n-dimensional Hilbert space spanned by the computational basis states of n quantum bits (qubits) involved in the program. The individual unitary transformations forming the quantum program are the quantum gates. A widely used approach to measure the complexity of a quantum circuit (a quantum program) is in terms of the number of elementary gates involved in the circuit [15] (or alternatively the depth of the circuit composed by these elementary gates).

In current quantum devices gates are still somewhat noisy due to undesirable interactions with the environment. Thus, by increasing the number of quantum gates in a circuit makes it harder to retrieve any meaningful result from the quantum computer. Therefore the ability to decompose quantum programs into low complexity gate arrays plays an important role in compiling programs onto currently available noisy quantum computers. Also, we need to keep in mind that some hardware components of a quantum computer is experimentally more demanding to realize than others. Depending on the underlying architecture [16] the operation error characteristic for one-qubit and two-qubit gates might differ even by an order of magnitude [17]. Thus, it is a commonly used strategy of decomposition algorithms to focus only on making the number of two-gates as low as possible in the resulting quantum circuit. Ref. [18] showed that the theoretical lower limit N_{theo} of $CNOT$ gates sufficient to decompose any n-qubit unitary is given by

$$N_{theo}(n) = \lceil \tfrac{1}{4}\left(4^n - 3n - 1\right) \rceil \,. \tag{1}$$

However, no circuit construction strategy has been so far developed that meets this lower limit of $CNOT$ gates.

A conventional, QR decomposition based approach [15] to implement general multi-qubit gates in the decomposition yields an array of $O(n^3 4^n)$ elementary quantum gates. Later the method was improved by employing Gray code basis [19] and a decomposing algorithm requiring $8.7 \cdot 4^n$ $CNOT$ gates [20] was developed. An analogous algorithm was developed by Shende et al [21] improving the limit of $CNOT$ gates to $2 \cdot 4^n - (2n+3)2n + 2n$ with iterative state preparation cycles to decompose a general unitary column by column.

Concurrently, an alternative method to decompose general unitaries was developed by the work of

Refs. [22] and [23]. The recursive algorithm based on the CSD method [24] makes it possible to decompose a general unitary with $\frac{1}{2}\cdot n\cdot 4^n - \frac{1}{2}\cdot 2^n$ $CNOT$ gates [22]. By further optimizations this limit can be lowered to $\frac{1}{2}\cdot 4^n - \frac{1}{2}\cdot 2^n - 2$ $CNOT$ gates [23]. The developed recursive CSD method is implemented in the Qubiter package [25] maintained up to these days. Currently the most efficient decomposing algorithm was introduced in Ref. [21]. The Quantum Shannon Decomposition (QSD) follows similar strategy than the recursive CSD method, just using eigenvalue decomposition during the recursive steps instead of the CSD method. In each iteration step the decomposition is applied to unitary gates containing of one less qubit than in the previous step. The QSD needs $\frac{3}{4}\cdot 4^n - \frac{3}{2}\cdot 2^n$ $CNOT$ gates to decompose a general unitary, which limit can be lowered to $\frac{23}{48}\cdot 4^n - \frac{3}{2}\cdot 2^n + \frac{4}{3}$ $CNOT$ gates by additional optimizations [21]. Recently an efficient implementation of the QSD method developed in the OpenQL [26] package was published in Ref. [27].

Despite of the great progress in the field, the described decomposing algorithms suffer from a crucial limitation. Namely, they assume full connectivity between the qubits during the decomposition. However, to run quantum algorithms on currently available gate-model based quantum computers, quantum circuits must be adopted to the constraints of the underlying hardware. Reference [23] reported on a modified CSD method adopted to an architecture with only nearest neighbour connections between the qubits. Here the limitations of the connectivity were incorporated into the building blocks of the decomposing quantum circuit. Using this modified algorithm the $CNOT$ gate count was increased by a factor of less than two compared to the case of full connectivity. Another strategy to compile a quantum circuit on a realistic quantum device is to insert additional $SWAP$ operations to bring qubits close to each other to apply two-qubit controlled gates on them. To further decrease the gate count hardware specific strategies can be developed to determine the order of the individual $SWAP$ operations [28, 29]. However, the resulting gate count seems to be still to large and versatile error-correction, mapping or scheduling measures need to be applied during the post processing of the quantum computational results [30, 26, 26].

In this work we propose a novel strategy to approximately decompose general unitaries into quantum circuits consisting of single- and two-qubit controlled quantum gates. Our numerical method is based on an iterative, sequential optimization (SO) of the parameters of the incorporated quantum gates making it possible to support hardware specific limitations of the target device, without the need of any post-processing or addition of extra $SWAP$ gates. In an ideal case of full connectivity between the qubits our algorithm requires $CNOT$ gate count very close to $N_{theo}(n)$ in order to decompose a general quantum program. In particular, 15 $CNOT$ gates turned to be sufficient to decompose a general 3-qubit unitary with high numerical precision, while the decomposition of a 4-qubit unitary requires only 63 $CNOT$ gates compared to 20 and 100 $CNOT$ gates needed by the optimized QSD, the most efficient algorithm known today. We note that all the previously mentioned algorithms – including the SO method – aim to capture the fewest possible number of $CNOT$ gates being sufficient to decompose any unitary. Besides this "worst case" scenario, the quest to find the most efficient decomposition of individual quantum programs is frequently addressed problem in research works [31, 32, 33, 34, 35]. For example, in Ref. [34], published during the preparation of this paper, the authors proposed a strategy to optimize both the single qubit rotations and the gate structure by eliminating trivial single-qubit gates and simplifying the remaining gate structure using $CNOT$ gate identities. This method enabled them to achieve significant compression for some of the studied quantum circuits, while for the majority of the addressed circuits the compression turned to be less pronounced.

In this work, we focus on the decomposition of general unitaries by considering random unitary matrices representing the worst-case scenario, and we leave the study of special quantum programs for a future work. Although we can't provide a rigorous argument to decide whether the SO approach is mathematically exact, our numerical results do not exclude this possibility. We also tested our algorithm to compile general 3- and 4-qubit quantum programs for the well known 5-qubit device QX2 [36] of IBM and for the heavy hexagonal lattice design implemented in the falcon quantum processors [37] and aimed to be used in forthcoming generations of IBM quantum processors as well. The decomposition of the addressed quantum programs for these architectures needed only about 13% more $CNOT$ gates than in the ideal full connectivity case. The proof-of-principle implementation of our algorithm [called Sequential Quantum Gate Decomposer (SQUANDER)] is accessible via GitHub [38].

The rest of the paper is organized as follows. In Sec. 2 we describe the details of the SO algorithm. Then in Sec. 3 we numerically analyse the performance of the SO algorithm implemented in the SQUANDER package to decompose general 3 and 4-qubit unitaries in case of full connectivity design. Section 4 provides a detailed report on how to perform gate decomposition implementing custom quantum gate types and adopt the decomposition process to sparse connectivity designs. Finally, in Secs. 5 and 6 we provide implementation details of the SQUANDER package and summarize the main results of our work, respectively.

2 Description of the heuristic algorithm

The central idea of our heuristic algorithm was inspired by the logic of neural networks, where a preconstructed structure of function nodes is trained on data sets to mimic some specific function behaviour. In our case the function nodes in the network are chosen from a universal set of one- and two-qubit gates, which can be considered as linear functions acting on specific qubit states of the quantum register. The computing nodes are connected via quantum wires (representing the timeline of the qubits) and via the action of two-qubit gates. The initially separated qubits become entangled due to the action of a general quantum program represented by a unitary U (see Fig. 1). Since the final purpose of the gate network is to get the qubits disentangled from each other, the trained gate structure would be the decomposition of $U^{-1} = U^{\dagger}$, and the training procedure would be the process of the decomposition.

A fundamental difference compared to machine learning is, however, that in order to decompose a unitary, we need to approach the global minimum of the cost function very precisely. Secondly, each unitary U would need an individual optimization process to find a suitable minimum of the cost function (that is sufficiently close to the global minimum), so the optimization process must be very efficient since it needs to be performed every time we want to decompose a quantum program. Also, the theoretical lower limit of the $CNOT$ gates necessary to decompose general unitaries scales with $\sim 4^n$, so the number of the free parameters associated with the single-qubit gates surrounding the $CNOT$ gates would also increase rapidly with the number of qubits involved in the problem. In summary, the optimization process needs to be efficient and it should be able to find a close-to-global minimum in a large parameter space at the same time. These requirements poses a serious challenge on the optimizing algorithm. Surprisingly, later in this section we will see that it is possible to design such an algorithm, even if it is specific to our problem, and it highly depends on some peculiar properties of the gates incorporated into the gate design. Now we turn our attention to the construction of the universal gate structure being later trained to decompose quantum programs.

2.1 Description of the gate structure

Our heuristic algorithm follows the strategy to disentangle qubits from the others sequentially, i.e. one by one. The disentanglement of a single qubit is achieved by an application of a systematic gate structure on the qubits similarly to the technique reported in *Theorem 9* of Ref. [21] used for state preparation. To this end we generalized the concept of multi-controlled gates [39] by applying single-qubit rotations to both the target and control qubits, as can be seen in Fig. 1 for the case of an $n = 3$ qubit register. By introducing additional degrees of freedom into the system, one might expect that fewer $CNOT$ gates would be necessary to disentangle a qubit from the others. On the other hand, it also becomes harder to determine the optimized parameters of the $U3$ gates. Since it is not possible to obtain these parameters by a deterministic procedure we determine them by a recursive optimization process discussed later in Sec. 2.3. (We note, that a $CNOT$ gate can be expressed using another $CNOT$ gate with reversed orientation and with four Hadamard gates acting on the control and target qubits before and after the reversed $CNOT$ gate, see Fig.2. Hence, the orientation of the $CNOT$ gates in Fig. 1 would not play an important role.) In order for easier understanding, we group the elementary single- and two-qubit gates into layers according to the pair of qubits on which they apply transformation. For example, the first layer in Fig. 1 has a direct effect on qubits q_1 and q_2, while the second layer acts on qubits q_0 and q_2. The generalized gate structure used to disentangle a qubit from the others follows the general logic that (i) each layer consists of two $U3$ gates and one controlled two-qubit gate acting on a pair of qubits; (ii) the gate structure is constructed from periodically repeating sequences of the layers; (iii) each layer must involve the qubit that we want to disentangle from the others; (iv) each of the remaining qubits must be present exactly once in the shortest periodically repeating sequence of the layers.

The rules (i)-(iv) fully determine the number of layers in the shortest period of the gate structure, while the order of the layers inside one period might be arbitrary. The total number of the layers in the design is a function of the number of the involved qubits and it also depends on the precision of the decomposition we want to achieve in a given time limit. In Sec. 2.3. we will provide detailed numerical analysis of our method and discuss how much of layers (or equivalently $CNOT$ gates) are necessary to disentangle a qubit from the remaining $n - 1$ ones.

Finally, we notice that the efficiency of our decomposing algorithm does not depend on the specific orientation of the individual $CNOT$ gates. Thus, we arrange the $CNOT$ gates in the gate design such that the control qubit is always the qubit we are about to disentangle from the others. In addition, the $CNOT$ gates in the design can be replaced by other two-qubit controlled gates (although with some limitations to be discussed later), such as CZ (controlled rotation around the z axis) or CH (controlled Hadamard) gates.

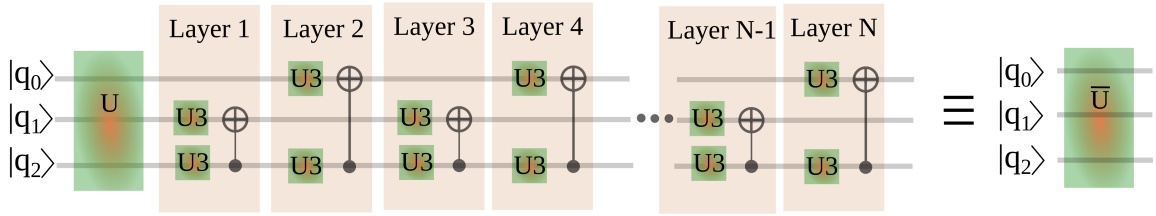


Figure 1: The gate structure composed from $U3$ and $CNOT$ gates used in the decomposition of a three-qubit unitary U. The $U3$ gates are represented by green boxes, while the $CNOT$ gates are expressed by vertical lines between two qubits. The gate structure applied after the quantum circuit U disentangle qubit q_2 from the qubits q_0 and q_1. The elementary gates are grouped into layers that are individually optimized one after another until the global minimum of the cost function f_{sub} is not reached up to a given accuracy. The unified quantum circuit composed from the unitary U and the operation layers 1 to N can be formally described by a quantum circuit denoted by $\overline{U}$



Figure 2: Expressing a $CNOT$ gate in terms of another $CNOT$ gate of reversed orientation and four Hadamard gates acting on both the control and target qubits.

2.2 The cost function of the optimization process

As we described in previous sections, the goal of the gate structure depicted in Fig. 1 (with appropriately chosen parameters of the $U3$ gates) is to disentangle the qubit q_2 from the others. The initially separable system of three qubits becomes entangled due to the action of the general three-qubit unitary U. The combined action of the unitary U and the decomposing gate structure can be expressed by a "joined" unitary $\overline{U}$, which is the product of U and the other gate operations present in the system. In general, the unitary $\overline{U}$ shown at the right side of Fig.1) can be organized in a block form

$$\overline{U} = \begin{pmatrix} \overline{U}_{00} & \overline{U}_{01} \\ \overline{U}_{10} & \overline{U}_{11} \end{pmatrix} , \tag{2}$$

where submatrices $\overline{U}_{ij}$ $(i,j \in \{0,1\})$ describe the dynamics of qubits q_0 and q_1, provided that qubit q_2 undergoes a transition from state $|j\rangle$ to state $|i\rangle$. If q_2 is independent from the rest of the qubits, then submatrices $\overline{U}_{ij}$ differ only by a constant factor from each other, and

$$\overline{U}_{ij} \cdot \overline{U}^{\dagger}_{pq} = \kappa^{pq}_{ij} \mathbb{1} , \tag{3}$$

where $\mathbb{1}$ is the identity matrix and κ^{pq}_{ij} is the upper left element of the product matrix $\overline{U}_{ij} \cdot \overline{U}^{\dagger}_{pq}$ calculated for indices $i,j,p,q \in \{0,1\}$. The cost function f_{sub} quantifying the entanglement of qubit q_2 with the rest of the qubits can be then defined as the "distance" of the submatrices $\overline{U}_{ij}$ from the equality given by Eq. (3):

$$f_{sub}(\overline{U}) = \sum_{l,m=1}^{2^{(n-1)}} \sum_{ijpq \in (0,1)} \left| \left(\overline{U}_{ij} \overline{U}^{\dagger}_{pq} - \kappa^{pq}_{ij} \mathbb{1} \right)_{lm} \right|^2 \tag{4}$$

In Eq. (4) the indices l,m label the rows and columns of the matrix enclosed by the parentheses. The upper limit of these indices is determined by n labeling the qubit that we are currently disentangling from the others. (In our specific case depicted in Fig. 1 $n=2$, since qubit q_2 is about to be disentangled from the rest.)

By summing up the squared norm of the matrix elements in Eq. (4) we end up with a single number characterizing the entanglement of qubit q_n with the rest of the qubits. The cost function f_{sub} reaches a global minimum 0 if Eq. (3) holds, i.e. qubit q_n is independent from the other ones. By reaching the global minimum of f_{sub} over the free parameters of the $U3$ gate operations we end up with a gate structure that is tuned to disentangle the qubit q_n from the others.

The described procedure can be applied for an arbitrary number of qubits, however, the total number of layers in the decomposing gate design is expected to increases exponentially with n. Before we present our numerical results we describe an efficient optimization method to solve the $f_{sub} \to 0$ optimization problem.

2.3 Solving the optimization problem

In this section we provide an efficient method to solve the optimization problem $f_{sub} \to 0$. The number of gate layers necessary to decompose the unitary U might vary from problem to problem, but depending on the number of qubits involved in the quantum program there is a maximal number of gate layers N involved in the decomposing gate structure that turns to be sufficient to approximate any N-qubit unitary U with high numerical accuracy. However, with the number of the applied layers, the number of free parameters of the optimization problem also increases.

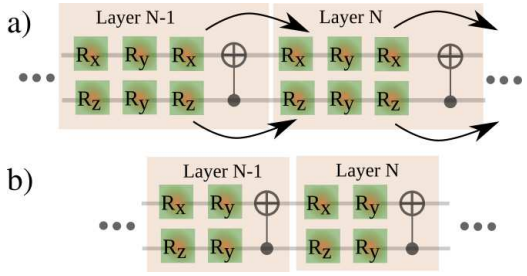


Figure 3: The rotations around z and x axis in subfigure a) can be merged into the successive layer of gate operations. Consequently, each layer can be parametrized with 4 parameters.

According to the reasoning of Ref.[40], rotations around the z and x axis commute with $CNOT$ gates on the control and target qubits, respectively. Since a general single-qubit rotation can be expressed by the combinations $R_zR_yR_z$ and $R_xR_yR_x$, one of the R_z/R_x gates can be merged into the successive layer as indicated in Fig. 3. Consequently, it is sufficient to keep only two free parameters per $U3$ operation, and eventually we end up with 4 free parameters per layer to be optimized. Despite of this reduction of the free parameters, the optimization problem $f_{sub} \to 0$ still poses a serious challenge to be solved. Fortunately, it turns out that the problem can be solved in an iterative way by grouping the layers into blocks that can be individually optimized, while the parameters of the other blocks are kept constant. After we found the conditional minimum of the cost function f_{sub} by optimizing the parameters of a given block, we continue the procedure by optimizing the parameters of the following block and we repeat this step over and over again. In each iteration step we optimize only a single block of layers and keep the rest of the free parameters constant. Following this sequential procedure (also inspiring the naming of the SO algorithm) the cost function of the optimization problem gets smaller in each iteration step until it converges to a global minimum.

Although we can not provide a rigorous mathematical explanation, the outlined block-wise optimization procedure showed a very robust tendency in approaching the global minimum of the cost function f_{sub}. We believe that these exceptional properties of the model are related to the presence of the two-qubit controlled gates in the design. The sequence of the abrupt transformations (related to the controlled gates) on the initial input state induces large changes in the parameter landscape of the upcoming layers when the parameters of the optimized block are changed. Thus, the block-wise optimization of the system has a self-reinforcing tendency to explore large areas in the parameter space favouring to find a spot corresponding to a global minimum.

We tried the same block-wise optimization procedure using parametrized two-qubit operations, such as controlled rotation gates. In these gates the free parameter is the angle of the controlled rotation making the abrupt state transformation less pronounced. In this case the state transformation over the sequence of the layers becomes much smoother and the optimization process rather converges to a higher local minimum instead of a global one.

Finally we notice, that the size of the computational blocks can be chosen arbitrary, as far as they remain small enough to be tractable via classical algorithms, such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm[41] used in our implementation. Also, since the performance of the outlined algorithm highly depends on the time-scale of a single-block optimization, it is a good practice to keep the size of the blocks low.

2.4 Decomposition of general unitaries

In previous sections we introduced the main ideas to construct a gate design to disentangle a single qubit from the others and formulated the central mathematical problem to be solved to this end. The outlined procedure can be repeated recursively to disentangle all of the qubits one after another. In each recursive step a new unitary $U_{\text{new}} = \overline{U}_{ij}$ is chosen from the subblocks of the previously disentangled $\overline{U}$ and the gate structure is extended by new decomposing layers in order to disentangle another qubit from the others. Since the subblocks $\overline{U}_{ij}$ differ from each other only by a scalar factor, the choice of the subblock is arbitrary. Figure 4. shows the gate structure used to fully decompose a general three-qubit unitary U. The first $N-3$ layers are used to disentangle qubit q_2 from the rest and layers $N-2$, $N-1$ and N are responsible to disentangle the remaining two qubits q_1 and q_0. (To decompose a general two-qubit unitary one needs not more than 3 $CNOT$ gates [42, 18]) After all the qubits became independent from each other, we can drive them into their initial state by applying individual $U3$ rotation on them. At the end of the algorithm we perform a final fine tuning on the free parameters of the full decomposing quantum circuit in order to get as close as possible to the synthesised unitary. In this optimisation process we use the metric based on the Frobenius norm defined in Refs.[33, 34] to quantify the distance between the initial and the approximated unitary. Strictly speaking, the constructed quantum circuit provides the inverse of the initial quantum program U up to a constant phase factor. Since mathematically the inverse of a unitary is defined by it's complex transpose, the gate structure we have just constructed can be interpreted as the gate decomposition of $U^\dagger$.

We notice, that in our implementation of the algorithm we perform the disentanglement of the qubits in a specific order starting with the qubit with the highest label and continue with qubits in descending order

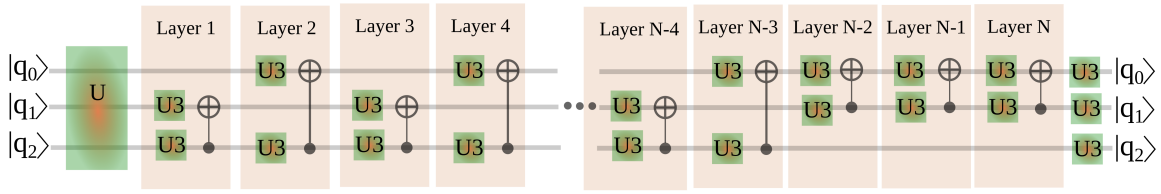


Figure 4: The scheme of the full gate structure reverting the qubits q_i $[i \in (0, 1, 2)]$ into their initial separable state. The qubit q_2 is being disentangled from the others by the first $N-3$ layers. The remaining two qubits are disentangled via the operations of layers $N-2$, $N-1$ and N. Finally, all of the independent qubits are rotated into their initial state.

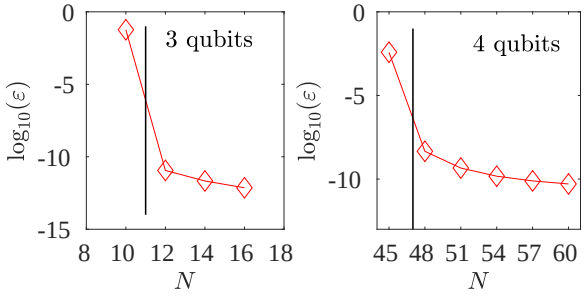


Figure 5: The numerically obtained averaged minimum of the cost function f_{sub} defined as $\varepsilon = \min(f_{sub})$ in terms of the number N of the implemented decomposing layers. $\overline{\varepsilon}$ was computed as the average of 10 independent runs of minimizing f_{sub} with a new random unitary in each run. The vertical black lines represent the theoretical lower bound $N_{theo}(n) - N_{theo}(n-1)$ needed to disentangle the n-th qubit from the rest. As one can see, $\overline{\varepsilon}$ starts to rapidly increase below the theoretical lower bound, while it saturates above this limit.

of labels. In order to perform the decomposition in different order of the qubits one need to reorder the list of the qubits. In Sec. 4.2 we provide a specific use case where the reordering of the qubits is needed to decompose a unitary for a specific hardware with connection limitations between the qubits.

3 Numerical results

In the forthcoming sections we examine the numerical properties of the SO algorithm. The most important question is to determine the necessary resources needed to decompose a general quantum program into a sequence of quantum logical gates. Before turning our attention to the decomposition of a general unitary, we first examine haw many decomposing layers are needed to disentangle a single qubit from a register of entangled n qubits. Although our algorithm is not limited to specific two-qubit gates, we start the discussion of our numerical results obtained when $CNOT$ gates were implemented in the gate structure. The analytic expression $N_{theo}(n)$ provides a starting point to determine the number of layers needed to disentangle a qubit from the others. In particular, we estimate the minimal number of $CNOT$ gates to $N_{theo}(n) - N_{theo}(n-1)$. (The decomposition of an n-qubit unitary needs at least $N_{theo}(n)$ $CNOT$ gates, which can be split between the disentanglement of the first qubit from the others and the decomposition of the remaining $(n-1)$-qubit unitary needing $N_{theo}(n-1)$ $CNOT$ gates.) Here we examine how close we can optimize the parameters of f_{sub} to obtain the target minimum of 0 needed to disentangle a qubit from the rest. If the number of the layers (which is equal to the number of two-qubit controlled gates) is larger than the theoretical lower limit, we expect that the free parameters of f_{sub} can be optimized to come close to the $f_{sub} \to 0$ solution with some numerical precision ε. As soon as we decrease the number of the layers below the theoretical lower limit, the global minimum of f_{sub} should rise above 0, since there would not be sufficient $CNOT$ gates in the circuit to meet the desired solution. Figure 5. shows the average value of $\varepsilon = \min(f_{sub})$ in terms of the applied number N of the decomposing layers. As we see in the figure, ε picks up large values below the theoretical lower limit indicated by black vertical lines. In this regime the optimization problem $f_{sub} \to 0$ has no solution. Above the theoretical limit the solution of the optimization problem saturates on $\varepsilon \leq \mathcal{O}(10^{-8})$ which can not be further decreased due to accumulated numerical errors during the calculations. The numerical precision of the calculations is limited by the double precision format of floating point number representation implemented in BLAS libraries[43]. Since matrix multiplication is the most intensively used operation during the optimization process, it has the greatest influence on the final numerical error. In principle, it would be possible to implement matrix multiplication using higher precision number representation in exchange of rapid increase of the execution time. However, the 64 bit architecture of modern CPU-s poses hardware limitation to do efficient matrix multiplication using higher precision number format. (We notice, that CPUs still provide floating point operation unit for extended precision numbers with 80 bit registers, but these hardware units do not support vectorization of floating point operations.)

At the end, one needs to make a trade-off between the final precision of the decomposition and the computational time. Our numerical analysis showed that double precision floating point number format is suf-

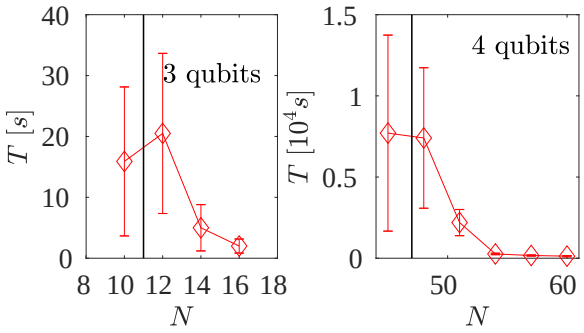


Figure 6: The execution time to find the global minimum of the cost function f_{sub}. The execution time was measured by averaging 10 independent runs to disentangle a qubit from the others as a function of the implemented layers. The standard deviance of the measured execution times is represented by the errorbars. The vertical black lines represent the theoretical lower bound $N_{theo}(n) - N_{theo}(n-1)$ needed to disentangle the n-th qubit from the rest. The numerical simulations were done on a traditional desktop PC with *Intel Core i7* CPU and with 8 GB of RAM.

ficient to achieve $\|U - U_{approx}\|_2 \sim \mathcal{O}(10^{-5})$ precision in the decomposition of general 3, 4 and 5-qubit unitaries on a reasonable time-scale, while the number of implemented $CNOT$ gates is very close to the theoretical lower limit. Here U is the initial unitary and U_{approx} is the quantum program standing for the decomposed quantum circuit. The norm $\|A\|_2$ stands for the spectral norm of matrix A, which corresponds to the largest singular value of A, i.e., the square root of the largest eigenvalue of the matrix $A^\dagger A$, where $A^\dagger$ denotes the conjugate transpose of A. (We note that U_{approx} might differ by a global phase factor from the initial unitary U. This global phase factor should be removed when calculating the error of the decomposition.)

The execution time necessary to find the global minimum of f_{sub} highly depends on the number N of the implemented layers in the circuit. Figure 6. shows the execution time of the optimization process in terms of the applied number N of decomposing layers. As one can see the execution time highly increases as the number of layers comes close to the theoretical lower limit indicated by the black vertical lines. By increasing the number of the free parameters in the problem by adding extra decomposing layers, the convergence to the global minimum of 0 becomes much faster, so it becomes computationally easier to decompose a unitary by increasing the number of $CNOT$ gates. Another option to reduce the execution time is to set a hard limit on ε, and when the optimization process reaches the limit ε_0, the implementation of the algorithm steps forward to disentangle the next qubit. By setting an optimization limit ε_0 to 10^{-8} the final error of the decomposition is turned to be as small as $\|U - U_{approx}\|_2 \sim \mathcal{O}(10^{-5})$ and $\mathcal{O}(10^{-4})$ for a general 3 and 4-qubit unitary, respectively. At the same time, the execution time of the decomposition decreases to several seconds for 3-qubit unitary incorporating 15 $CNOT$ gates in total and to 2 minutes in case of a 4-qubit unitary incorporating 75 $CNOT$ gates in total. (The execution time was measured on an *Intel Core i7* CPU having 8 threads.) However, these execution times are still not competitive with current implementations of the CSD or QSD implementations[27], for larger problems than 4 qubits, the decomposition time exceeds the limit of an acceptable time scale. Still, the low $CNOT$ gate count close to the theoretical lower limit might justify the increased execution time in many use-cases. For example, as we will see in Secs. 4 and 4.2, the SO algorithm is also applicable to optimize a gate structure on architectures with specific properties or limitations. In addition, the developed SO algorithm can be mixed with QSD to optimize the number of gates by letting the SO algorithm to decompose small unitaries generated during the QSD procedure[27].

In Table 1 we summarize our numerical results on unitary decomposition by reporting the lower bound of $CNOT$ gates needed to decompose a general unitary and we compare our results to other algorithms. Our studies revealed, that any three-qubit unitary can be approximated using at most 15 $CNOT$ gates by the SO algorithm. In addition, 63 and 267 $CNOT$ gates are sufficient to decompose a 4 and 5-qubit unitary, respectively up to precision of $\mathcal{O}(10^{-5})$ and $\mathcal{O}(10^{-4})$ for 3 and $4-5$ qubit unitaries, calculated with the spectral norm metric defined earlier in the text.

4 Architecture specific decomposition

The most challenging hardware components of quantum computers are the two-qubit controlled gates that are used to induce entanglement between the qubits. Depending on the underlying architecture, the nature of the interaction between the qubits might suit for the creation of different kinds of two-qubit gates. For example, in Ref. [44] a microwave irradiation could be used to form a cross resonant gate between the qubits, Refs. [45, 46] reported on experimental demonstration of controlled-phase gates between transmon qubits, while another realization of controlled-phase gates between two photonic qubits encoded in photonic fields stored in cavities or between time-bin qubits were accomplished in Refs. [47] and [48], respectively. Usually, the $CNOT$ gates in these platforms are generated as a composite gate operation formed from one two-qubit contolled gate and from suitable one-qubit rotations[46, 48].

Another typical issue on quantum hardware is related to limitations in inter-qubit connectivity. Since two-qubit gates can be in many experimental situations realized only between qubits close to each other a gate decomposition of a quantum program needs to

Number of qubits	2	3	4	5	n
Iterative unentangling [21]	8	62	344	1642	$2 \cdot 4^n - (2n+3) \cdot 2^n + 2n$
Givens rotations [20, 27]	4	64	536	4156	$\approx 8 \cdot 4 \cdot 4^n$
Recursive CSD [22]	14	92	504	2544	$\frac{1}{2} \cdot n \cdot 4^n - \frac{1}{2} \cdot 2^n$
Recursive CSD (optimized) [23]	4	26	118	494	$\frac{1}{2} \cdot 4^n - \frac{1}{2} \cdot 2^n - 2$
QSD [21, 27]	6	36	168	720	$\frac{3}{4} \cdot 4^n - \frac{3}{2} \cdot 2^n$
QSD (optimized)[21]	3	20	100	444	$(23/48) \cdot (4^n) - \frac{3}{2} \cdot 2^n + \frac{4}{3}$
Sequential optimization[a]	**3**	**15**	**63**	**267**	Not available
Theoretical Lower Bounds[18]	3	14	61	252	$\frac{1}{4}(4^n - 3n - 1)$

Table 1: The number of $CNOT$ gates needed to decompose a general unitary for the individual algorithms. The sequential optimization implemented in the SQUANDER package provides the lowest number of $CNOT$ gates needed for the decomposition $\mathcal{O}(10^{-5})$ and $\mathcal{O}(10^{-4})$ for 3 and $4-5$ qubit unitaries, calculated with the spectral norm metric defined in Sec. 3.

[a]The numbers given in the table represent the limiting case of the sequential optimization algorithm resulting in the highest execution time. The decomposition would be faster by orders when more $CNOT$ gates are used for the decomposition. For further details see the main text.

be adopted to these limitations. In most cases this is accomplished by adding extra $SWAP$ gates into the circuit, which tends to highly increase the gate cunt. However, in order to increase the total fidelity of a quantum circuit, it would be more suitable to develop a decomposition algorithm that natively supports the limitations of the targeted quantum hardware. In this section we show that the SO algorithm can be adopted to such needs.

4.1 Custom two-qubit controlled gates in the decomposition

As we mentioned earlier in Sec. 2.3, the SO algorithm is not limited to specific choice of the gates used in the decomposition. The decomposition can be executed with arbitrary, but parameter-less two-qubit controlled gates as well.

The SQUANDER package (for details see Sec. 5.) provides a Python (and native C++) interface to construct custom gate structure for the decomposition. In Alg. 1. we provide a brief overview on how to define such custom gate structure to disentangle qubit $q2$ form qubits $q0$ and $q1$ in a three-qubit problem. It is sufficient to define only a single period of the gate structure, since this period is repeated during the decomposition automatically. (The number of periods in the decomposing gate design can be set by an individual parameter. For details see the tutorial material of the SQUANDER package[49].) First we import the necessary package from SQUANDER and define the number of the qubits by a local variable. Then we create an instance of class *qgd_Gates_Block* representing one period in the decomposing gate structure. The individual layers *Layer1* and *Layer2* of the gate design are represented by the same type of class as the period itself. Finally we add the two- and one-qubit gates to the layers using the methods implemented in the class *qgd_Gates_Block*. We chose to add controlled phase (CZ) gate to the first layer and controlled Hadamard gate to the second layer. One period of the constructed gate structure is shown in Fig.7. Equivalently to the case when only $CNOT$ gates were implemented in the design, there is needed at least 12 decomposing layers to disentangle qubit $q2$ from the others. Since the implemented two-qubit gates are related to the $CNOT$ gate by one-qubit rotations, this equivalence is straightforward.

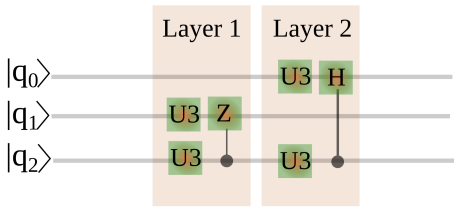


Figure 7: A period of a custum gate structure used to disentangle qubit 2 from qubits 0 and 1 in a three-qubit problem. Qubit 2 is connected to qubit 1 via controlled pahse gate, while the interaction with qubit 0 is realized via controlled Hadamard gate. The source code to generate such gate structure is described in Alg. 1.

The SQUANDER package contains a working example implementing the described use-case, it can be also found in the tutorial material at [49]. We notice, that the code example in Alg. 1. shows only the steps to construct custom gate structure, to implement it in actual gate synthesis additional steps are needed that are described in the tutorial material as well.

4.2 Adopting decomposition to a hardware specific connectivity structure

All numerical results presented in previous sections were obtained for designs without any limitations

Algorithm 1: Source code example using the Python interface of the SQUANDER package to construct a period of decomposing gate structure with custom two-qubit gates in the design.

output: A period in the three-qubit gate structure used to disentangle qubit $q2$ from the rest.

```
from qgd_python.gates.qgd_Operation_Block import qgd_Operation_Block

def CustomGates():
    # The number of the qubits in the register
    qubit_num = 3

    # creating an instance of class representing a period of the gate structure
    Gate_Period = qgd_Gates_Block( qbit_num )

    # create the first decomposing layer with CZ gate
    Layer1 = qgd_Gates_Block( qbit_num )

    # add U3 gates acting on qubits 0 and 1 with two free parameters (Θ and λ) to Layer1
    Theta = True
    Phi = False
    Lambda = True
    Layer1.add_U3( 1, Theta, Phi, Lambda )
    Layer1.add_U3( 2, Theta, Phi, Lambda )

    # add CZ gate to layer1 with control qubit 2 and target qubit 0
    Layer1.add_CZ( 1, 2)

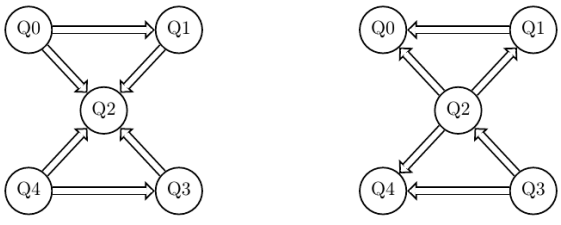
    # create the second decomposing layer with controlled Hadamard gate
    Layer2 = qgd_Gates_Block( qbit_num )

    # add U3 gates acting on qubits 1 and 2 with two free parameters (Θ and λ) to Layer2
    Theta = True
    Phi = False
    Lambda = True
    Layer2.add_U3( 0, Theta, Phi, Lambda )
    Layer2.add_U3( 2, Theta, Phi, Lambda )

    # add controlled Hadamard gate to layer2 with control qubit 2 and target qubit 1
    Layer2.add_CH( 0, 2)

    # add the decomposing layers to the period of the decomposing gate structure
    Gate_Period.add_Gates_Block( Layer1 )
    Gate_Period.add_Gates_Block( Layer2 )
    return Gate_Period
```

regarding the connections between the individual qubits, i.e. each of the qubit pairs could be connected by two-qubit gates. However, in realistic experimental setups only neighbouring qubits can be connected in a direct gate operation. In many cases even the orientation of the two-qubit controlled gates are fixed, it is not possible to chose the control or the target qubits on demand. Examples of such architecture are represented in Fig. 8 showing the case of the 5-qubit quantum devices of IBM publicly available via IBM Cloud services[50]. The ability to decompose a quantum program into gate structures with sparse connectivity is undoubtedly an important issue of quantum programming. One theoretically possible way to overcome this



QX2[36] QX4[51]

Figure 8: Connectivity graph between the qubits in the QX2 and QX5 five-qubit IBM quantum computers [29]. A qubit shown at the tail of the arrows can only work as control qubit and the qubit shown at the head of the arrow can only work as target qubit.

issue is to apply swap gates to get distant qubits close to each other where the desired two-qubit controlled operation can be applied on them, and then move the qubits back to their initial position by another swap operations. In order to increase the fidelity of a quantum circuit constructed this way, there have been proposed several optimization approaches [28, 29] that might considerably reduce the number of gate operations in a quantum program. Efficient gate reduction techniques are becoming even more important as the number of qubits continuously increase on the hardware side, while the connectivity density stays low. On the 8-th of August, 2021, IBM announced to implement their future quantum devices basing on the so-called heavy hexagonal lattice design, which offers even less connection between the qubits than previous architectures.

Here in this section we report on a novel approach to optimize a quantum circuit of general unitaries. The key point of our approach is to find the best decomposition of a quantum program from the start, without implementing additional swap gates into the decomposed circuit. To this end we use the SO method on custom gate structure adopted to the architecture of the target hardware. Our method gives the lowest possible number of $CNOT$ gates in the decomposition of a general unitary, close to the theoretical lower limit. As an example we show how to decompose a 4-qubit quantum program on the QX2 5-qubit device of IBM[36].

The choice of the four qubits might be in principle arbitrary, as long as the chosen set contains the central Q_2 qubit. In our example we decompose a general 4-qubit unitary on qubits Q_0, Q_1, Q_2 and Q_3 of the QX2. At the beginning of the decomposition we need to decide the order in which we are going to disentangle the qubits from the others. In general, qubits having the most connections are the easiest to disentangle from the others. However, since qubit Q_2 plays a central role in the design, we need to choose another qubit to begin with, otherwise there would be left no direct connection to qubit Q_3 and it would be not possible to finish the decomposition without using swap gates. For example, we might choose qubit Q_0 as the first one to disentangle. (We notice that at this point qubit Q_1 would be equivalently good choice.) Then we can continue the decomposition with getting qubit Q_1 independent from the others, and finally we disentangle the remaining two qubits Q_2 and Q_3. Since the SQUANDER package process the decomposition in specific order of the qubits (always disentangling the qubit with the highest index), we need to re-label the qubits, so SQUANDER can disentangle the qubits in the correct order.

Figure 9.a) shows the relabeled qubits used in the decomposition. In order to construct a suitable gate design to disentangle qubits $q3$ and $q2$, we need to abandon rule (iii) formulated in Sec. 2.1 for the ideal

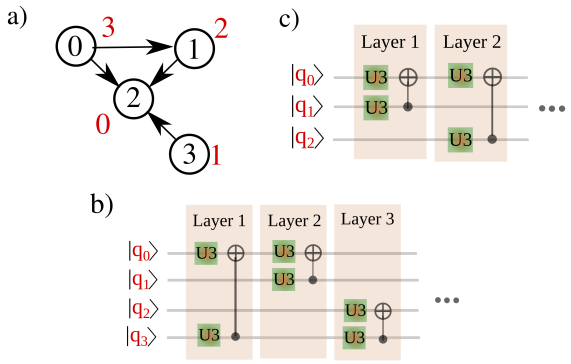


Figure 9: a) Re-labeling the qubits of the original QX2 architecture to perform a decomposition in correct order. The new labels used by the SQUANDER package are indicated by the red numbers, while the black numbers corrspond to the original labeling of the qubits. b) and c) The periods of the gate structures to disentangle qubits 3 and 2 from the rest, respectively. Here the qubits follows the labeling indicated by the red numbers in a).

full connectivity case. Since qubit $q3$ does not have direct connection to all the remaining qubits, it has no point to involve this qubit in all the decomposing layers when disentangling it. Instead, as shown in Fig. 9.b), we mediate the interaction between qubits $q3$ and $q1$ (which are not directly connected) via qubit $q0$, since $q0$ is connected to both $q3$ and $q1$. Similarly, when it comes to disentangle qubit $q2$, we need to use again qubit $q0$ to mediate between $q2$ and $q1$, as shown in Fig. 9.c). Due to these mediated interactions between the qubits one might expect that more $CNOT$ gates would be needed for the decomposition as in the case of ideal full connectivity. Unfortunately, there is no rule to tell how many decomposing layers would be needed, we know only the lower limit of the $CNOT$ gates from Table. 1, which is a good starting point to begin with. Our numerical results shown in Fig. 10 indicate that for the QX2 architecture we indeed need more decomposing layers to disentangle qubits from the rest. Surprisingly, the precision ε of the disentanglement comes very close to the full connectivity case by adding only several extra decomposing layers to the circuit. To disentangle the 4-th qubit in a four-qubit register one needs 54 layers (i.e. 54 $CNOT$ gates) to achieve the same precision as in the full connectivity case, while in the three-qubit case 14 layers are sufficient to get the same precision. In overall, the SO algorithm needs at least 17 $CNOT$ gates to decompose a general three-qubit unitary on the QX2 architecture and achieving the same precision as in the full connectivity case, while 71 $CNOT$ gates are sufficient in the 4-qubit case.

In heavy hexagonal structure of the qubits the connectivity between the qubits is even more reduced in order to increase the scalability of the architecture for fabrication. Here each unit cell of the lattice consists of a hexagonal arrangement of qubits, with an addi-

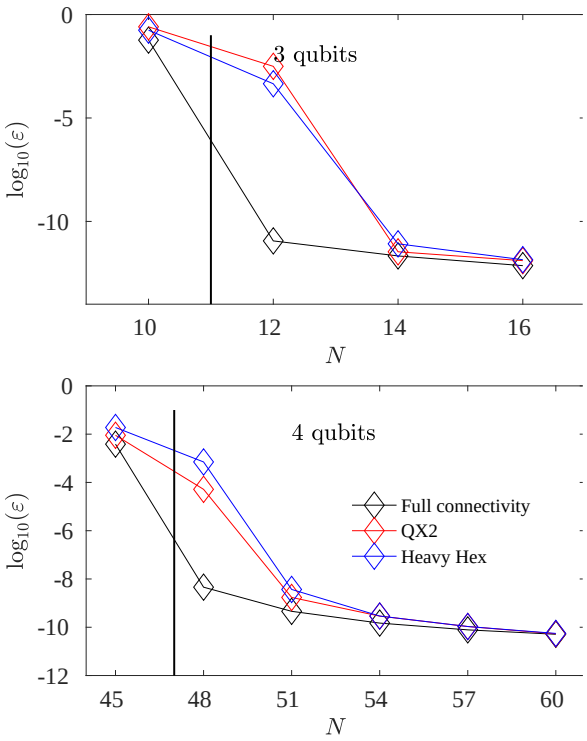


Figure 10: The numerically obtained average minimum of the cost function f_{sub} defined as $\varepsilon = \min(f_{sub})$ in terms of the number N of the implemented decomposing layers defined by the design in Figs. 9 and 11 for QX2 and heavy hexagonal architectures, respectively. $\overline{\varepsilon}$ was computed as the average of 10 independent runs of minimizing f_{sub} with a new random unitary in each run. The standard deviance of the calculated mean is smaller than the size of the data markers. The vertical black lines represent the theoretical lower bound $N_{theo}(n) - N_{theo}(n-1)$ needed to disentangle the n-th qubit from the rest. As one can see, the reduced connectivity between the qubits in QX2 and heavy hexagonal architectures implies more decomposing layers needed to disentangle a qubit.

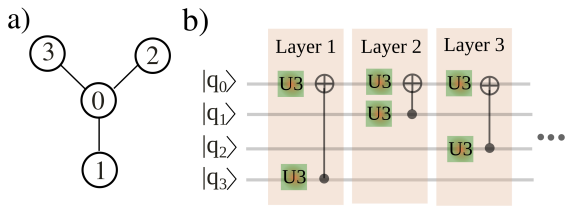


Figure 11: a) Selecting a 4 qubit sub-system in a heavy hexagonal lattice for a 4-qubit qunatum program. b) A period of a gate structure to disentangle qubit 3 from the others.

tional qubit on each edge. IBM first introduced the so-called "heavy hex" topology in 2020, the family of falcon quantum processors already implemented this design[37]. According to IBM the key idea behind reducing the number of connections between the qubits is justified by minimizing both the qubit frequency collisions and spectator qubit errors. However, less connectivity makes quantum circuits harder to implement. Figure 11.a) shows a 4-qubit "piece" of the heavy hex structure. Even if the $CNOT$ gates are (in principle) bidirectional in these designs, the missing connection between qubits 3 and 2 might increase the depth of quantum circuits compared to the QX2 architecture studied previously. Surprisingly, our numerical results showed that by using the gate structure depicted in Fig. 11.b) to disentangle qubit 3 from the others, there is no need for more $CNOT$ gates as were used for the QX2 architecture. According to Fig. 10, the obtained minima of f_{sub} are very close to each other for the QX2 and for the heavy hexagonal architectures, indicating that the missing connection between qubits 2 and 3 does not play a crucial role in the decomposition. It turns out that to decompose a general 4-qubit (or 3-qubit) unitary one needs the same number of $CNOT$ gates on both the QX2 and the heavy hexagonal architectures. Comparing this with the case of a fully connected topology, it is surprising how small the CNOT-count overhead is. It would be very interesting to study how this overhead would change for the heavy hexagonal lattice when we increase the number of qubits.

5 SQUANDER: implementation of the sequential optimization algorithm

Generally, a numerical procedure to solve an optimization problem is computationally more demanding than a deterministic or an iterative algorithm used to solve linear algebraic problems. For this reason, our implementation is not expected to be competitive in speed with other well known decomposition implementations, such as was reported in Ref. [27].

Still, in order to increase the computational performance of our implementation of the sequential optimization algorithm the solver engine was implemented in native C++/C programming language. In principle the execution time of the code can be further decreased by combining dense and sparse matrix operations during the optimization process. In order to keep the benefits coming with the flexibility and the popularity of a high level Python API we also designed a Python interface for our implementation, enabling to export the results of the decomposition into the well known quantum programming package Qiskit. (Interfacing with other quantum packages are under development.)

Our implementation, the Sequential Quantum Gate

Decomposer (SQUANDER) package is accessible through a public GitHub repository [38], while a documentation of the package is hosted at *CodeDocs[xyz]* site [49]. The SQUANDER package is equipped with the Threading Building Block library [52] providing an efficient task oriented parallel programming model to achieve an optimal workload balance among the accessible execution units of the underlying hardware avoiding any over-subscription of the resources. The register level parallelism via portable SIMD instruction are provided by the incorporation of low level BLAS functions.

6 Conclusion and Outlook

In this work, we constructed and numerically analyzed an optimization method to approximate general unitaries by quantum circuit made of one- and two-qubit gate operations. At this point our approach is based on numerical heuristics, it is hard to make any mathematically rigorous arguments about the limits of the algorithm. Solving the optimization problem by a sequential technique enables one to get close to the global minimum of the cost function f_{sub} up to an accuracy hitting the numerical precision accumulated by the floating point operations. The approach works for arbitrary combination of two-qubit controlled gates and gives the lowest number of the two-qubit controlled gates necessary to decompose a general unitary. (For details see Table. 1.) However, it is still an open question whether the developed numerical approach can be considered to be an exact solution or not.

We also examined the possibility of using our approach to decompose quantum programs on architectures having sparse connectivity between the qubits. Our method natively supports architecture specific features of the underlying architecture. The decomposition methods like the CSD or QSD, on the other hand, assume ideal, all-to-all connectivity between the qubits. In order to adopt a decomposed quantum circuit to a specific architecture, one needs to include additional swap gates and perform post optimization strategies to reduce the depth of the resulting quantum circuit. In the case of the sequential optimization algorithm, however, the decomposition can be adopted to hardware specifics from the start, resulting in a significantly lower depth of the quantum circuit.

Unfortunately, the increased execution time needed to find the optimized parameters of the gate structure poses a serious limitation on the applicability of our algorithm. The decomposition of a 7-qubit unitary takes more than a week, which limits the use-cases of the sequential decomposition algorithm to low number of qubits. In order to make use of the sequential optimization algorithm in the decomposition of larger unitaries, our algorithm can be combined with other decomposing approaches. In particular, QSD follows a similar strategy than our algorithm. By disentangling qubits one after another, in each iterative step a new unitary is produced with a less qubits than in the previous iteration step. By applying our algorithm at the end of this procedure – where 3, 4 or 5-qubit unitaries needs to be processed – one might significantly reduce the number of gates in the decomposition and make steps to adopt the decomposed quantum program to the connectivity specifics of the underlying hardware at the same time. However, we leave the study of this opportunity for future work. We believe that our results might trigger new strategies to turn quantum programs into efficient quantum circuits.

7 Acknowledgements

The research was supported by the Ministry of Innovation and Technology and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary, and was also supported by NKFIH through the Quantum Technology National Excellence Program (No.2017-1.2.1-NKP-2017-00001) and Grants No. 2020-2.1.1-ED-2021-00179, K124152, FK135220, KH129601. We acknowledge the computational resources provided by the Wigner Scientific Computational Laboratory (WSCLAB) (the formerWigner GPU Laboratory).

References

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

[2] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett.*, vol. 79, pp. 325–328, Jul 1997. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.79.325

[3] L. M. K. Vandersypen, M. Steffen, M. H. Sherwood, C. S. Yannoni, G. Breyta, and I. L. Chuang, "Implementation of a three-quantum-bit search algorithm," *Applied Physics Letters*, vol. 76, no. 5, pp. 646–648, 2000. [Online]. Available: https://doi.org/10.1063/1.125846

[4] C. Figgatt, D. Maslov, K. A. Landsman, N. M. Linke, S. Debnath, and C. Monroe, "Complete 3-qubit grover search on a programmable quantum computer," *Nature Communications*, vol. 8, no. 1, p. 1918, Dec 2017. [Online]. Available: https://doi.org/10.1038/s41467-017-01904-7

[5] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, "Exper-

imental realization of shor's quantum factoring algorithm using qubit recycling," *Nature Photonics*, vol. 6, no. 11, pp. 773–776, Nov 2012. [Online]. Available: https://doi.org/10.1038/nphoton.2012.259

[6] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt, "Realization of a scalable shor algorithm," *Science*, vol. 351, no. 6277, pp. 1068–1070, 2016. [Online]. Available: https://science.sciencemag.org/content/351/6277/1068

[7] M. Amico, Z. H. Saleem, and M. Kumph, "Experimental study of shor's factoring algorithm using the ibm q experience," *Phys. Rev. A*, vol. 100, p. 012305, Jul 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.100.012305

[8] M. P. Harrigan *et al.*, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Physics*, vol. 17, no. 3, pp. 332–336, 2021.

[9] F. Arute *et al.*, "Hartree-fock on a superconducting qubit quantum computer," *Science*, vol. 369, no. 6507, pp. 1084–1089, 2020.

[10] A. Smith, M. S. Kim, F. Pollmann, and J. Knolle, "Simulating quantum many-body dynamics on a current digital quantum computer," *npj Quantum Information*, vol. 5, no. 1, p. 106, Nov 2019. [Online]. Available: https://doi.org/10.1038/s41534-019-0217-0

[11] S. Leontica, F. Tennie, and T. Farrow, "Simulating molecules on a cloud-based 5-qubit ibm-q universal quantum computer," *Communications Physics*, vol. 4, no. 1, p. 112, Jun 2021. [Online]. Available: https://doi.org/10.1038/s42005-021-00616-1

[12] D. A. Fedorov, M. J. Otten, S. K. Gray, and Y. Alexeev, "Ab initio molecular dynamics on quantum computers," *The Journal of Chemical Physics*, vol. 154, no. 16, p. 164103, 2021. [Online]. Available: https://doi.org/10.1063/5.0046930

[13] T. Bian and S. Kais, "Quantum computing for atomic and molecular resonances," *The Journal of Chemical Physics*, vol. 154, no. 19, p. 194107, 2021. [Online]. Available: https://doi.org/10.1063/5.0040477

[14] K. Satzinger *et al.*, "Realizing topologically ordered states on a quantum processor," *arXiv preprint arXiv:2104.01180*, 2021.

[15] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.52.3457

[16] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Experimental comparison of two quantum computing architectures," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3305–3310, 2017. [Online]. Available: https://www.pnas.org/content/114/13/3305

[17] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 987–999. [Online]. Available: https://doi.org/10.1145/3297858.3304007

[18] V. V. Shende, I. L. Markov, and S. S. Bullock, "Minimal universal two-qubit controlled-not-based circuits," *Phys. Rev. A*, vol. 69, p. 062321, Jun 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.69.062321

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in FORTRAN. The art of scientific computing*, 1992.

[20] J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, "Efficient decomposition of quantum gates," *Phys. Rev. Lett.*, vol. 92, p. 177902, Apr 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.92.177902

[21] V. Shende, S. Bullock, and I. Markov, "Synthesis of quantum-logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1000–1010, 2006.

[22] R. R. Tucci, "A rudimentary quantum compiler(2cnd ed.)," 1999.

[23] M. Möttönen and J. J. Vartiainen, "Decompositions of general quantum gates," 2005.

[24] C. Paige and M. Wei, "History and generality of the cs decomposition," *Linear Algebra and its Applications*, vol. 208-209, pp. 303–326, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0024379594904464

[25] R. T. H. Dekant, H. Tregillus and T. Yin, "Qubiter at github," 2020. [Online]. Available: https://github.com/artiste-qb-net/qubiter

[26] N. Khammassi, I. Ashraf, J. v. Someren, R. Nane, A. M. Krol, M. A. Rol, L. Lao, K. Bertels, and C. G. Almudever, "Openql : A portable quantum programming framework for quantum accelerators," 2020.

[27] A. M. Krol, A. Sarkar, I. Ashraf, Z. Al-Ars, and K. Bertels, "Efficient decomposition of unitary matrices in quantum circuit compilers," 2021.

[28] G. W. Dueck, A. Pathak, M. M. Rahman, A. Shukla, and A. Banerjee, "Optimization of circuits for ibm's five-qubit quantum computers," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 680–684.

[29] M. Sisodia, A. Shukla, A. A. A. de Almeida, G. Dueck, and A. Pathak, "Circuit optimization for ibm processors: A way to get higher fidelity and higher values of nonclassicality witnesses," *arXiv: Quantum Physics*, 2018.

[30] M. Rötteler, *Quantum Error Correction*. Boston, MA: Springer US, 2008, pp. 705–708. [Online]. Available: https://doi.org/10.1007/978-0-387-30162-4_315

[31] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, "Quantum-assisted quantum compiling," *Quantum*, vol. 3, p. 140, May 2019. [Online]. Available: https://doi.org/10.22331/q-2019-05-13-140

[32] E. Younis, K. Sen, K. Yelick, and C. Iancu, "QFAST: Quantum Synthesis Using a Hierarchical Continuous Circuit Space," *arXiv e-prints*, p. arXiv:2003.04462, Mar. 2020.

[33] E. Younis, K. Sen, K. Yelick, and C. Iancu, "Qfast: Conflating search and numerical optimization for scalable quantum circuit synthesis," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2021, pp. 232–243. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/QCE52317.2021.00041

[34] L. Madden and A. Simonetto, "Best approximate quantum compiling problems," 2021.

[35] E. Smith, M. G. Davis, J. Larson, E. Younis, C. Iancu, and W. Lavrijsen, "Leap: Scaling numerical optimization based synthesis using an incremental approach," 2021, arXiv:2106.11246.

[36] "Ibm q5 yorktown." [Online]. Available: https://github.com/Qiskit/ibmq-device-information/tree/master/backends/yorktown/V1

[37] P. Jurcevic, A. Javadi-Abhari, L. S. Bishop, I. Lauer, D. F. Bogorin, M. Brink, L. Capelluto, O. Günlük, T. Itoko, N. Kanazawa, A. Kandala, G. A. Keefe, K. Krsulich, W. Landers, E. P. Lewandowski, D. T. McClure, G. Nannicini, A. Narasgond, H. M. Nayfeh, E. Pritchett, M. B. Rothwell, S. Srinivasan, N. Sundaresan, C. Wang, K. X. Wei, C. J. Wood, J.-B. Yau, E. J. Zhang, O. E. Dial, J. M. Chow, and J. M. Gambetta, "Demonstration of quantum volume 64 on a superconducting quantum computing system," *Quantum Science and Technology*, vol. 6, no. 2, p. 025020, mar 2021. [Online]. Available: https://doi.org/10.1088/2058-9565/abe519

[38] "Sequential quantum gate decomposer," 2021. [Online]. Available: https://github.com/rakytap/sequential-quantum-gate-decomposer

[39] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Quantum circuits for general multiqubit gates," *Phys. Rev. Lett.*, vol. 93, p. 130502, Sep 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.93.130502

[40] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, "Quantum circuits for isometries," *Physical Review A*, vol. 93, no. 3, Mar 2016. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.93.032318

[41] C. Kelley, *Iterative Methods for Optimization*, ser. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1999. [Online]. Available: https://books.google.hu/books?id=Bq6VcmzOe1IC

[42] F. Vatan and C. Williams, "Optimal quantum circuits for general two-qubit gates," *Phys. Rev. A*, vol. 69, p. 032315, Mar 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.69.032315

[43] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry *et al.*, "An updated set of basic linear algebra subprograms (blas)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.

[44] J. M. Chow, A. D. Córcoles, J. M. Gambetta, C. Rigetti, B. R. Johnson, J. A. Smolin, J. R. Rozen, G. A. Keefe, M. B. Rothwell, M. B. Ketchen, and M. Steffen, "Simple all-microwave entangling gate for fixed-frequency superconducting qubits," *Phys. Rev. Lett.*, vol. 107, p. 080502, Aug 2011. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.107.080502

[45] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, no. 7497, pp. 500–503, Apr 2014. [Online]. Available: https://doi.org/10.1038/nature13171

[46] Z. Zong, Z. Sun, Z. Dong, C. Run, L. Xiang, Z. Zhan, Q. Wang, Y. Fei, Y. Wu, W. Jin, C. Xiao, Z. Jia, P. Duan, J. Wu, Y. Yin, and G. Guo, "Optimization of a controlled-z gate with data-driven gradient-ascent pulse engineering in a superconducting-qubit system," *Phys. Rev. Applied*, vol. 15, p. 064005, Jun 2021.

[Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.15.064005

[47] Y. Xu, Y. Ma, W. Cai, X. Mu, W. Dai, W. Wang, L. Hu, X. Li, J. Han, H. Wang, Y. P. Song, Z.-B. Yang, S.-B. Zheng, and L. Sun, "Demonstration of controlled-phase gates between two error-correctable photonic qubits," *Phys. Rev. Lett.*, vol. 124, p. 120501, Mar 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.124.120501

[48] H.-P. Lo, T. Ikuta, N. Matsuda, T. Honjo, W. J. Munro, and H. Takesue, "Quantum process tomography of a controlled-phase gate for time-bin qubits," *Phys. Rev. Applied*, vol. 13, p. 034013, Mar 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.13.034013

[49] "Tutorial materials to use sequential quantum gate decomposer," 2021. [Online]. Available: https://codedocs.xyz/rakytap/sequential-quantum-gate-decomposer/

[50] "Ibm quantum solutions." [Online]. Available: https://www.ibm.com/quantum-computing/

[51] "Ibm q5 tenerife." [Online]. Available: https://github.com/Qiskit/ibmq-device-information/tree/master/backends/tenerife/V1

[52] M. Voss, R. Asenjo, and J. Reinders, *Pro TBB: C++ parallel programming with threading building blocks*. New York: Apress Open, 2019. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-4398-5