IOP Publishing New J. Phys. 27 (2025) 104509 https://doi.org/10.1088/1367-2630/ae0e40

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

PAPER

Optimization driven quantum circuit reduction

Bodo Rosenhahn[1,*], **Tobias J Osborne**[2] and **Christoph Hirche**[1]

[1] Institute for Information Processing (tnt/L3S), Leibniz Universität Hannover, Hannover, Germany
[2] Institute for Theoretical Physics (ITP/L3S), Leibniz Universität Hannover, Hannover, Germany
[*] Author to whom any correspondence should be addressed.

E-mail: rosenhahn@tnt.uni-hannover.de

Keywords: quantum circuits, transpilation, machine learning

OPEN ACCESS

RECEIVED
10 April 2025

REVISED
26 August 2025

ACCEPTED FOR PUBLICATION
1 October 2025

PUBLISHED
22 October 2025

Abstract

Implementing a quantum circuit on specific hardware with a reduced available gate set is often associated with a substantial increase in the length of the equivalent circuit. This process is also known as transpilation and due to decoherence, it is mandatory to keep quantum circuits as short as possible, without affecting functionality. In this work we propose three different transpilation approaches, based on a localized term-replacement scheme, to substantially reduce circuit lengths while preserving the unitary operation implemented by the circuit. The first variant is based on a stochastic search scheme, and the other variants are driven by a database retrieval scheme and a machine learning based decision support. We show that our proposed methods generate short quantum circuits for restricted gate sets, superior to the typical results obtained by using various qiskit and Berkley quantum synthesis toolkit optimization levels. Our method can be applied to different gate sets and scales well with an arbitrary number of qubits.

1. Introduction

As the realization of a fully fault-tolerant quantum computer capable of supporting error-free logical qubits and gates draws ever nearer, it is vital that methods to optimize the length of quantum circuits be investigated. The reasons here are twofold: (1) just below the fault tolerance threshold decoherence will still play a critical role and shorter quantum circuits will suffer less; and (2) just above threshold the implementation of a logical gate is likely to still be very expensive in terms of physical qubits and gates. Thus the development of schemes to discover optimal quantum circuits are a priority.

The task of translating, or *transpiling*, a given quantum circuit into a sequence of gates realizable on a specific hardware architecture is known as *quantum architecture mapping* (QAM) [11, 57]. Realizing QAM brings about four key challenges, (a) *gate synthesis*, to convert gates or gate sequences into a set of gates supported by the target hardware; (b) *gate mapping*, to rearrange the qubits in the circuit to match the connectivity constraints of the target hardware; (c) *gate optimization*, to minimize the overall number of gates in the circuit; and (d) *the resource allocation*, to ensure that the circuit does not exceed the resources available on the target hardware.

The automated search for optimal quantum circuits to implement a target unitary is also known as *quantum architecture search* (QAS). The name is motivated by terminology arising in the machine learning (ML) community, where *neural architecture search* [34, 66] deals with algorithm selection and its hyper-parameter tuning. Common approaches are based on reinforcement learning, structural search or performance prediction, as presented in [3, 8, 36].

QAS is often based on discrete and heuristic optimization strategies as the optimization criteria can be non-differentiable. Here, for example, Gibbs sampling [27], evolutional approaches [16], genetic algorithms [26, 47], combinatorial search [2], neural-network based predictors [70], variants with noise-aware circuit learning [10], and the optimization of approximate solutions [72] have been suggested. Several works also demonstrated that gradient-descent based optimization schemes [63, 71], Monte

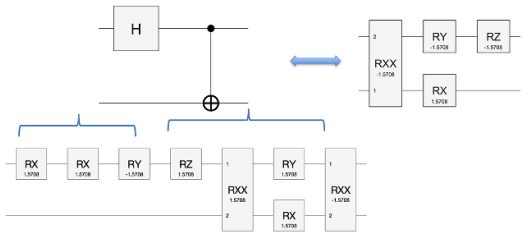

Figure 1. Motivation: the quantum circuit to produce a Bell state (H-Gate followed by a CNot Gate). Left: naive quantum architecture mapping to an ion trap quantum computer gate set, by mapping one operator after another. Right: resource efficient (and equivalent) quantum circuit.

Carlo tree search (MCTS) [62], Monte Carlo graph search (MCGS) [50, 51], ranking schemes [19], random coordinate descent [15] and reinforcement learning [64] are promising strategies. Recent surveys on QAS are provided in [30, 73].

Often, QAS, QAM, or transpilation, involves methods which have been motivated from electronic design automation [59]. It should also be noted, that these terms are often used inconsistently in the literature. Indeed, terms like QAS as terminology to perform quantum circuit optimization (even if it is using ML tools) can be quite misleading.

There have been a variety of papers investigating quantum circuit optimization. In [11] the two-dimensional square, heavy-hex and fully hexagonal qubit coupling lattices are considered as targets for transpiled quantum circuits. Implementing quantum algorithms on multi-core quantum computing architectures was discussed in [42] and [74], which present a methodology for mapping quantum circuits to the IBM-QX architectures. Brandhofer *et al* [5] address the primary objective of adapting a quantum circuit to the topology of a provided quantum computer so that the qubit–qubit interaction requirements of each computation are satisfied. To adapt to the topology, several SWAP gate insertion models are optimized using a Z3 SMT solver [14]. The authors carried out numerical experiments on quantum circuits involving up to 15 qubits and a maximum depth of 76. In [35] the architecture mapping for trapped ion qubits is addressed. Besides single traps, so-called quantum charge coupled device architectures allow the use of several traps in parallel. This poses optimization challenges for the shuttling, split/merge, and swap operations required to perform quantum computations. Most of these works rely on evolutionary optimization [12], reinforcement learning [28], and similar techniques as they are used in QAS. The present work is mainly inspired by [5, 51], but our focus is on circuit reduction and we therefore assume an existing mapping as an upper bound on the circuit length, and we propose the use of local term replacement operators to reduce the circuit length iteratively. Our method combines a random search strategy with local optimization and is method-wise inspired by several works from the field of stochastic optimization [31, 43, 56, 58, 69].

Several software packages have been provided to compile and transpile quantum software. For our experiments we mainly use qiskit [1] and Berkley quantum synthesis toolkit (BQSKit) [68]. Whereas qiskit is a well established and easy to use package, supported e.g. by IBM and its online tools, the BQSKit is a very powerful software, with gate deletion and rule-free gate transpilation algorithms. Several well known algorithms (leap, qsearch, qfast, qpredict, PAS and more) are released as part of BQSKit [10, 13, 54, 65, 67]. As BQSKit outperforms several commercial compilers (Qiskit, Cirq, Tket), we use it as additional baseline for our experiments.

A key challenge for transpilation is that only a handful of gates are provided by an existent hardware. Thus the naive translation of individual blocks, followed by the assembly of the blocks, one after the other, can lead to vastly long quantum circuits exhibiting many redundancies and inefficiencies. Due to decoherence and/or the overhead of quantum error correction, it is vital to find an efficient and shorter equivalent quantum circuit to obtain reliable results. This is especially important for, e.g. quantum volume metrics [52], that measure the capabilities and error rates of a quantum computer. Such metrics are typically used to evaluate and compare quantum computers across different platforms. Figure 1 illustrates the challenges arising in realizing QAM in terms of a small example to prepare a Bell state using elementary realizable quantum gates (left). An optimized circuit is depicted on the right.

In the classical literature the analogous problem to QAM is related to *logic minimization*, which addresses the task of replacing a group of (logic) gates with another (smaller group) that will perform

the same task faster or by using less space on a chip. A standard motivation is that the equivalent optimized circuit is cheaper, more compact on a chip, more energy efficient, has reduced latency, and it minimizes risks of cross-talk. Karnaugh maps [29] are a typical device to carry out such logic minimization. For Karnaugh maps, the idea is to order the constraints via a so-called Gray code. A Gray code ensures that only one variable changes between each pair of adjacent cells. The minimal terms for the final expression are found by encircling groups of 1 s in the Karnaugh map which provide the best opportunities to simplify the expression. Unfortunately, this approach can not be directly applied to quantum circuits. Still, the same arguments for logic minimization also hold for quantum circuit optimization: Short and stable quantum circuits are more energy efficient, have reduced latency and they are more stable as there is a minimized risk of decoherence.

To address the challenges of realizing large-scale automated quantum circuit reduction we propose, in this paper, the use of stochastic, database and ML supported reduction methods to identify redundancies and algebraic manipulations to efficiently reduce the circuit length. Our core contributions are three different methods to perform quantum circuit reduction, and the evaluation of their impact on quantum hardware:

1. Method 1 (V1-random sampling (RS)): we propose a simple sub-block selection and RS based term replacement algorithm to identify reducible circuit blocks in a quantum circuit.
2. Method 2 (V2-database retrieval (DR)): we study the precomputation of a compute graph of predefined depth to derive a database of optimal (yet still reasonably small) quantum maps. Then a random selected sub-block and database retrieval (DR) scheme is used to efficiently reduce these sampled circuit blocks.
3. Method 3 (V3-random forest (RF)) is built on V2, where we let a classifier make a decision if a sampled sub-circuit block is likely reducible. In our experiments, the decision is made by a RF, which is very fast to compute at inference time. Only after the classifier decision is the DR scheme called. This prevents unnecessary database look-ups which can be time consuming.
4. An evaluation on real quantum hardware reveals that the reduced quantum circuits have a significant impact on the results. Our approach scales up to an arbitrary amount of qubits (as explained in section 4.2) and is superior to various qiskit and BQSKit optimization levels.
5. The source code for the graph generation and quantum circuit length optimization will be made publicly available [38].

The remainder of this paper is structured as follows: section 2 starts with the preliminaries on quantum gates and circuits, summarizes compute graphs for get sets and provides some detail about RFs. Section 3 contains the proposed methods, starting with a RS based approach for quantum circuit reduction (V1), followed by the DR approach by using the compute graph (V2) and the RF supported database lookup (V3). The RF is used to decide whether it is worthwhile performing a database look up, or not and therefore reduces sampling inefficiency significantly. All methods build on each other and compensate effects arising from the baseline approach. Section 4 contains our experiments. We decided to perform experiments on two gate sets. The first gate set we use is common for ion-trap architectures and comprises RX, RY, RZ and RXX gates. The second gate set we use is common for NISQ-architectures and comprises RX, RZ and CZ gates. Section 5 summarizes our paper and gives a brief outlook to further work.

2. Preliminaries

2.1. Quantum gates and circuits

For the sake of simplicity we assume that a target quantum computer is comprised of N *logical* qubits, forming a quantum register [22]. Thus the Hilbert space appropriate for our system is given as $\mathcal{H} \equiv (\mathbb{C}^2)^{\otimes N} \cong \mathbb{C}^{2^N}$. The axioms of quantum mechanics posit that quantum logic gates are unitary matrices. Thus, a gate acting on N qubits is represented by a $2^N \times 2^N$ unitary matrix. A quantum gate sequence is then simply a set of such gates which are, in turn, evaluated via a series of matrix multiplications: A quantum circuit of length L is thus described by an ordered tuple $(O(1), O(2), \ldots, O(L))$ of quantum gates and the corresponding unitary operation U is given by the product

$$U = O(L)\,O(L-1)\cdots O(1)\,. \tag{1}$$

Standard elementary quantum gates often involve the Pauli-(X, Y, Z) operations, as well as Hadamard-, CNOT-, SWAP-, phase-shift-, and TOFFOLI-gates, all of which are expressible as standardized unitary

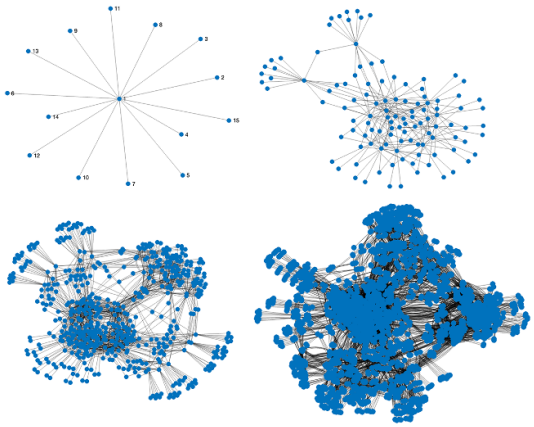

Figure 2. Compute graphs of depth 1–4 (from the upper left to the lower right) for a provided set of available quantum operators.

matrices [37]. The action of a quantum gate is extended to act on a register of any size by making use of a tensor product with the identity operator.

We interpret each quantum circuit as a token chain, e.g. $U = O(L)O(L-1)\cdots O(1) =$ `O(L) O(L-1) … O(1)`, where each token represents an operator of the quantum circuit. As some operators are commutative it is possible to swap specific tokens without changing the resultant unitary implemented by the quantum circuit; we will perform term replacement on this token chain to reduce the length of an existing quantum circuit without changing its functionality.

2.2. Compute graphs

If we are provided a limited set of available gates then it is possible to explore all possible implementable quantum circuits by building the so-called *compute graph* up to a predefined depth. These are combinatorial objects with nodes given by quantum circuits and edges labelled by elementary gates.

In our preliminary work [51] we made use of such a compute graph to perform quantum architecture optimization via MCGS. The method starts with the identity operator $\mathbb{I}$ as root node and iteratively builds up quantum circuits by selecting gates from a predefined set $\mathcal{OP} = \{O_1, O_2, \ldots\}$ of elementary quantum gates. The compute graph is then a graphical model with nodes containing unitary matrices and edges encoding an elementary unitary operator $O_i \in \mathcal{OP}$. The graph is initialized with the identity matrix $\mathbb{I}$ as root node. An operator O_i is selected and applied to the root node. This yields a new node by multiplying the selected operator with the unitary matrix of the parent node. If the resulting unitary already exists as node in the graph, a direct edge from the parent to the already existing node can be added. Otherwise, a new node is generated and connected with the parent node. Please note that we compare unitaries numerically with a tolerance of 10^{-5} and we compensated for a global phase. While growing the graph, the resulting unitary matrices are provided as graph nodes and the underlying quantum circuit can be computed by finding the shortest path from the root node to the target unitary and by collecting the operators along the edges of the path. Figure 2 shows the emerging graph with increasing depth and we refer to [51] for further details.

Thus, each node is identified with a possible quantum circuit. It is notable that this graph contains cycles since identical quantum circuits have multiple representations in terms of different gates and gate orders. As the graph will increase exponentially, it is only feasible to build up a full graph for reasonably short circuits with a limited gate set. For MCGS it is therefore important to restrict the growth direction in an unbalanced fashion to explore graph structures which are more likely to be useful for solving a certain task. Poisson sampling can exploited as the underlying sampling process to select a vertex to further develop the current compute (sub)graph. It is the basic paradigm of Monte Carlo search [33] and adapted Gibbs sampling [17] to iteratively grow a graph. In contrast to the former work, we fully grow the graph to a certain depth and collect for all nodes the shortest path to the root as the most efficient quantum circuit implementation of the node unitary. The table 1 summarizes how the graph increases with increasing depth. Note that several cycles in the developed graph exist, and for a pool of e.g. n operators, a depth of m leads to far fewer nodes than n^m. This is the main reason for the efficiency of MCGS compared to classical MCTS models. Figure 2 shows the compute graphs for 14 predefined operators along the depth 1, 2, 3 and 4. Note, that this graph can be pre-computed and resulting unitary matrices (on the nodes), as well as their perfect factorization, can be stored in a database. This will be

Table 1. Full compute graphs for different depths and numbers of operators.

Qubits	#Operators	Depth	Nodes	Edges
2	14	1	15	14
2	14	2	114	210
2	14	3	584	1596
2	14	4	2024	8176
2	14	5	4512	28 336
2	14	6	7420	63 168
3	24	1	25	24
3	24	2	337	600
3	24	3	3215	8088
3	24	4	23 622	77 160
3	24	5	137 572	566 928

later used (V2) to obtain a more efficient term replacement scheme compared with a stochastic random search.

2.3. RFs

Our last variant V3 is based on a fast machine-learning based decision. To elucidate this scheme we first summarize in this section the principle of a decision tree, and based on this, a RF. A decision tree is a hierarchical model performing splits based on local decisions. Given a training data set, the goal is to find a splitting criterion that maximizes the information gain by measuring the entropy of the parent (before the decision) and comparing it to the entropies of the children (after the decision). More formally, the goal of a split node is to maximize the information gain of the decision. We exploit the Shannon entropy, defined as

$$H(T) = I_E(p_1, p_2, \ldots, p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i, \tag{2}$$

where $p_1, p_2, \ldots$ are probabilities that add up to 1 that represent the percentage of each class present in the data set. Then the information gain of a splitting can be expressed as the sum of entropies of the children nodes subtracted from the entropy of the parent node. Thus, the information gain at a node T using attribute a can be measured as

$$\begin{aligned} \underbrace{IG(T,a)}_{\text{info gain}} &= \underbrace{H(T)}_{\text{Ent (parent)}} - \underbrace{H(T \mid a)}_{\text{Ent (children)}} \\ &= -\sum_{i=1}^{J} p_i \log_2 p_i + \sum_{i=1}^{J} \Pr(i \mid a) \log_2 \Pr(i \mid a) \end{aligned}$$

where $H(T|a)$ is the conditional entropy of T given the value of attribute a. The information gain is used to identify the perfect splitting nodes to build up a decision tree. We refer to [46] for further details.

Based on the decision scheme a RF can be generated as an ensemble of several (different) decision trees [6]. To ensure different decision trees, bootstrap aggregation and bagging is used, which means that for each decision tree only a random subset of the training data and available features is used and the final decision is based on the average (regression tree) or majority vote (classification tree) [7]. Please note, that the training of a RF is very fast, e.g. the most complex models we used in the below experiments were trained in less than a second on a simple notebook. The inference is much faster (below 0.0001 s).

3. Proposed methods

This section summarizes the token model to represent quantum circuits and is followed by three proposed methods for iterative quantum circuit reduction. Figure 3 provides a general overview of the method and indicates the three proposed variants.

3.1. Quantum circuits as token chain

The core idea is to represent a quantum circuit as a 1D token chain. Our method starts with an operator pool such as *RX*, *RZ* and *CZ* for a NISQ architecture. We describe this for the simple example of three

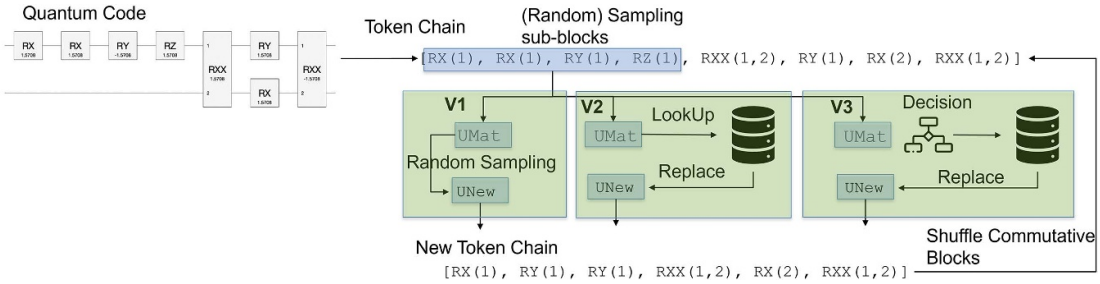

Figure 3. Variants of the quantum circuit optimization process: V1 is based on simple random sampling (RS), V2 is based on a database retrieval (DR) by using compute graphs and V3 is based on a decision scheme whether it is worth to perform the database lookup (RF).

qubits. Each gate acts on either one (e.g. a RX gate) or two qubits (e.g. a CZ gate) and takes eventually a parameter θ. Four discretized angles $[-\frac{\pi}{2}, -\frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{2}]$ are taken for the moment. For three qubits $[1,2,3]$ it is now possible to generate 12 gates, namely,

$$\begin{aligned}&\left[RX\left(1,-\tfrac{\pi}{2}\right),\ldots,RX\left(1,\tfrac{\pi}{2}\right),\right.\\&RX\left(2,-\tfrac{\pi}{2}\right),\ldots,RX\left(2,\tfrac{\pi}{2}\right),\\&\left.RX\left(3,-\tfrac{\pi}{2}\right),\ldots,RX\left(3,\tfrac{\pi}{2}\right)\right].\end{aligned}$$

Which are the tokens $1\ldots12$. We can use the tokens $13\ldots24$ to represent

$$\begin{aligned}&\left[RZ\left(1,-\tfrac{\pi}{2}\right),\ldots,RZ\left(1,\tfrac{\pi}{2}\right),\right.\\&RZ\left(2,-\tfrac{\pi}{2}\right),\ldots,RZ\left(2,\tfrac{\pi}{2}\right),\\&\left.RZ\left(3,-\tfrac{\pi}{2}\right),\ldots,RZ\left(3,\tfrac{\pi}{2}\right)\right].\end{aligned}$$

Finally, we can combine the CZ gates to

$$[CZ(1,2), CZ(1,3), CZ(2,3)].$$

For the tokens 25 ... 27. Since the CZ-gate is commutative across the qubits there is no need to generate e.g. $CZ(2,1)$ and this gate set results for three qubits and the provided angular discretization in an operator pool of 27 gates. The quantum circuit $RX(1,-\frac{\pi}{2})CZ(1,2)RZ(3,-\frac{\pi}{2})$ can then be expressed as the token chain [1, 25, 21]. Figure 4 shows on the left a quantum circuit and in the first line the resulting token chain.

3.2. V1: random search

As mentioned above, we interpret each quantum circuit as a token chain, e.g.

$$\begin{aligned}U &= O(L)\,O(L-1)\cdots O(1)\\&= \texttt{O(L) O(L-1) ...O(1)}\ ,\end{aligned}$$

where each token represents an operator of the quantum circuit. In theoretical computer science, a formal language consists of words whose letters are taken from an alphabet and are well-formed according to a specific set of rules called a formal grammar [20]. A formal grammar mainly consists of a set of production rules and rewriting rules for transforming strings. Each rule specifies a replacement of a particular string with another. Several books cover this fundamental topic, e.g. [61].

The random search algorithm works as follows: Given a token chain $U = O(L)O(L-1)\ldots O(1)$, first, a random connected subset, e.g. at position m of length n is selected,

$$\begin{aligned}U &= O(L)\ldots\underbrace{(O(m+n-1)\ldots O(m))}_{\text{length n}}\ldots O(1)\\&= O(L)\ldots(U_s)\ldots O(1).\end{aligned}$$

The subset $O(m+n-1)\ldots O(m)$ generates a unitary U_s. Then random sets of p tokens $O^r(i)$ with $p<n$ are sampled and the obtained unitary is compared with the unitary of the selected subset, $U_p =$

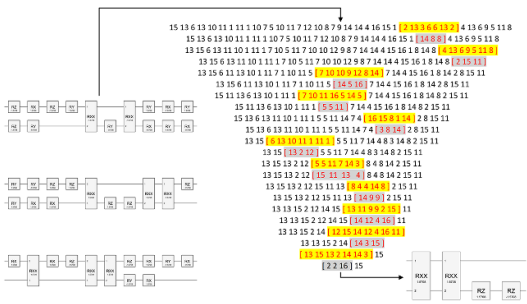

Figure 4. Example optimization steps of an input circuit length with 40 operators and intermediate reduction steps during the iterations until a substantially reduced circuit results.

$O^r(p)\dots O^r(1)$. If $U_p = U_s$ (up to a global phase and within a tolerance of 10^{-5}), the token chain can be replaced by a shorter one,

$$\begin{aligned} U &= O(L)\dots\underbrace{(O(m+n-1)\dots O(m))}_{\text{length}}n\dots O(1)\\ &= O(L)\dots\left(U_p\right)\dots O(1)\\ &= O(L)\dots\underbrace{(O^r(p)\dots O^r(1))}_{\text{length}}p\dots O(1)\,. \end{aligned}$$

Finally, commutative blocks are randomly exchanged and the process is iterated. Figure 4 shows the process for an input circuit with 40 gates on the left. These 40 gates have been randomly selected and translated to a token chain as described in the earlier sub-section. Then, from top to bottom, a term replacement procedure is shown. The yellow color indicates (also supported by the brackets) the selected token chain for reduction. The second (red) color shows the alternative token chain with a smaller amount of tokens (and gates). As the yellow tokens from the first line form the same unitary as the red tokens in the second line, the overall unitary of the whole quantum circuit remains unchanged. Then the process repeats with the remaining circuit till a convergence has been reached. The tiny token chain at the bottom of figure 4 is finally mapped back to the quantum circuit as shown in the lower right. Thus, the random search methods performs a localized term replacement of sub blocks of tokens without changing the overall unitary. In V1, both, the selection of the random tokens, as well as the search for a better token chain for term replacement are achieved using RS.

In general, RS is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability. This probability is increasing while more iterations are allowed. Random search algorithms have been proven useful for ill-conditioned global optimization problems, especially for problems where the optimization function can be nonconvex, nondifferentiable or discontinuous over a continuous, discrete, or mixed domain [4, 43, 53, 56]. On the other hand, random search strategies can lead to global optimal results (for an infinite time budget), but they suffer from sampling efficiency (iterated probing can take too long to find a good solution) and it is not guaranteed if a global optimal solution have been found [44]. To the best of our knowledge this is only possible for special optimization problems, e.g. expressible as a linear program in combination with branch and bound [9, 49]. The work [45] gives a theoretical justification of an optimality gap for approaches based in stochastic optimization, the work [55] provides a convergence proof. As our first method for quantum code transpilation involves two stochastic processes, the approach can become sampling inefficient. This motivates the two variants we present next.

3.3. V2: compute graph DR

Variant 2 is an extension of the random search strategy in V1. A major disadvantage of V1 lies in the two nested stochastic processes required to perform the random selection of the token chain to be optimized and the RS to find more efficient token chains. This can lead to a large computation time. One observation we made is that many blocks (e.g. of length five to eight) are often reduced to lengths between 2 and 4. This, in turn, is a circuit length which is feasible to fully compute within a compute graph, as presented in section 2.2. Thus, the idea is to collect all nodes with its corresponding unitary of

a compute graph and to compute its shortest path as its most efficient factorization and to store them in a database, e.g.

$$\begin{aligned} U_1 &\to O^1(n_1)\dots O^1(1)\\ U_2 &\to O^2(n_2)\dots O^2(1)\\ &\vdots\\ U_m &\to O^m(n_m)\dots O^m(1)\,. \end{aligned}$$

Table 1 shows how the size of this database (the amount of nodes) increases, even when only a small number of operators and a small depth is chosen. Still, e.g. a lookup and comparison with 20 K entries when there are 24 operators and a full depth of four is used, is manageable. More importantly, after the lookup, there is a guarantee whether the selected token chain can be reduced or not, which avoids unnecessary redundant compute steps.

As indicated in figure 3, the proposed V2 replaces the RS of subblocks with a database lookup. The experiments indicate that this can significantly reduce the computation time, see e.g. figure 5. Our approach can be further motivated and justified by earlier works combining local and global optimization [31, 32, 58].

3.4. V3: RF supported database lookup

The optimization version V2 relies on a database lookup. Unfortunately the database can become very large with an increasing number of operators and increasing depth (see table 1). Given a randomly selected token chain and the request to shorten it, we now propose to train a small classifier (here, we make use of a RF) to decide if it is worthwhile performing a database lookup. This classifier will be trained on the nodes of a compute graph to learn irreducible circuit blocks and successfully reduced circuit blocks from V2 as reducible examples. E.g. on our current implementation, the database lookup takes around 0.004 s, whereas a RF can be evaluated in 0.0001 s. Thus, a prefiltering based on the RF can prevent unnecessary computations and speed up the computation time. In a nutshell, if the RFs classifies a token chain as irreducible, this can be discarded and a new token chain can be selected. The recent work [45] provides a theoretical justification for the optimality and optimality gap of approaches based in stochastic optimization. They also show theoretically and empirically that the convergence can be accelerated by selecting sampling algorithms that cover the data set most effectively. Our approach can be seen as a variant of rejection sampling, which speeds up this step of selective sampling significantly [18, 23]. The optimization of a RF is summarized in section 2.3 and can be computed within seconds on a standard laptop. The RF is optimized beforehand and then used as existing method in our pipeline. Our software examples also provide the code for training a RF and is therefore fully reproducible. As indicated in figure 3, the proposed method V3 adds in front of the database lookup a small RF. We also experimented with small neural networks and alternative approaches like support vector machines. According to our experience, the RF offers a nice balance between efficiency and stability.

4. Experiments

In our experiments we will analyze the impact of our proposed algorithms from two sides. First, we will present the increased time efficiency of V2 and V3 compared to V1. On the second side, we will compare our proposed term replacement scheme with alternative transpilers (e.g. based on qiskit and BQSKit). Here, we show that our method can find more efficient quantum circuits due to the more complex algebraic manipulations which are performed with our approach.

4.1. Efficient quantum circuit reduction

This subsection demonstrates the efficiency of our proposed approaches to reduce a quantum circuit. The figure 1 shows a practical result for automated quantum circuit reduction as motivation. Thus, iterative term replacements can be used to reduce a quantum circuit to an equivalent one which uses less gates. We focus in our experiments on the few qubit regime (e.g. 4, 6 and 15 qubits), as here the correlation between the operators is more complex and intertwined than for a larger number of qubits. A large number of qubits leads to parts of the circuit becoming independent with increased number of qubits. Furthermore, several parts of the circuit are more likely to be irreducible. Therefore, the reduction of three or four qubit circuits can be more challenging and difficult than for a 20 qubit circuit block as more algebraic manipulations are required to efficiently reduce the blocks. We address the aspect of scaling to a larger qubit number in section 4.2, separately.

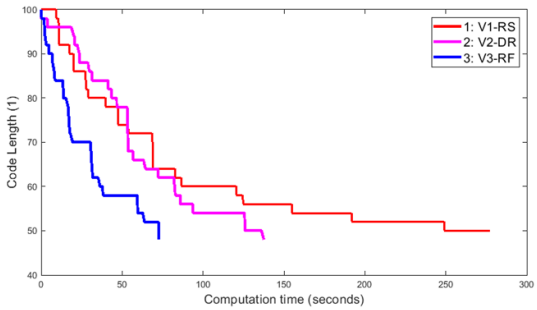

Figure 5. Quantum circuit reduction. The x-axis shows the computation time and the y-axis the circuit length which has to be reduced. The algorithm starts with a quantum circuit with 100 gates and with increasing time, the length is iteratively reduced to around 50 gates. As expected, the RS based approach is sampling inefficient, whereas the RF supported reduction scheme performs fastest.

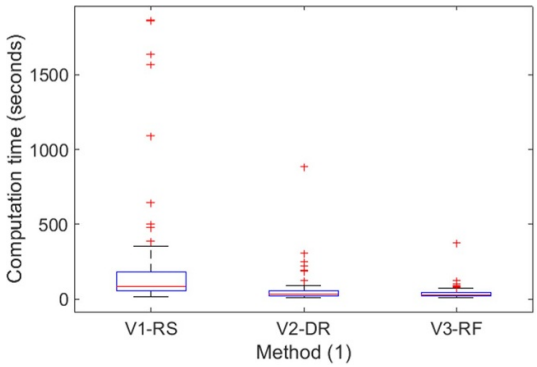

Figure 6. Statistical summary of the three proposed methods. 100 different randomly sampled quantum circuits are reduced to a predefined fixed length. The y-axis shows a box-plot. In each box, the central mark indicates the median, each box indicates the 25th and 75th percentiles. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the + marker symbol.

Table 2. Summarizing statistics on the computation time of the three proposed methods for 100 randomly generated quantum circuits, as in figure 6.

(100 runs)	V1-RS	V2-DR	V3-RF
Mean	199 (s)	55 (s)	38 (s)
Stddev	351.5	96.3	39.8

Figure 5 shows the performance of the three proposed algorithms on an example quantum circuit. In this case, the task is to reduce the circuit length of 100 to around 50. The x-axis shows the computation time in seconds and the y-axis the circuit length which is iteratively reduced during the iterations. The algorithm starts with a circuit of 100 gates (denoted as code length) at time point 0 and with increasing compute time, the gates are iteratively reduced to around 50 gates. As can be seen, all three proposed algorithms successfully converge, but the variants V2 and V3 are more efficient, with V3 being the fastest version. In our opinion, but not provable, a key of success is the global sampling strategy (random shuffling of commutative blocks and random sub block selection) with the local (perfect) optimization (using the compute graph) and a fast rejection scheme using the RF.

For our first set of experiments, we restrict the amount of qubits to up to 3. This implies, that all gates are highly connected and thus the optimization and algebraic manipulations to reduce the circuit are correspondingly harder. In section 4.2 we present an approach to upscale our method to an arbitrary number of qubits, while keeping the size of the compute graph constant.

As all three algorithms are based on a stochastic process (of random sub block selection), we further evaluated the performance in figure 6 and table 2. For this experiment we randomly generated 100 different starting circuits of length 100 and reduced the circuit to a target size. Then we measure the mean and standard deviation of all three variants over this large amount of repetitive experiments. As can be seen in table 2, the performance of the three proposed variants can be summarized as V3 performing as best algorithm (as the fastest one with the least standard deviation in results) and V1 performs worst.

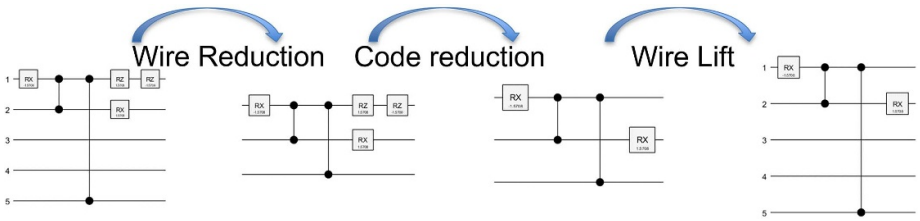

Figure 7. Principle for scaling up the number of qubits: since only short block segments are selected for reduction, non-used qubits an be removed, the optimization can be done on the lower dimensional space and the lifted up again to the original code size.

A statistical summary is also provided in figure 6 by using a box-plot for each method. The box-plot shows the median, the 25th and 75th percentiles as well as the most extreme data points (whiskers) and outliers. It should be noted, that we excluded the computation time to compute the graph and to train the small classifier as both parts can be done offline. Whereas V1-RS requires in average 199 s, already the database lookup reduces the compute time to nearly a quarter (55 s). The variant V3 making use of the RF brings the average computation time down to 38 s. Also the standard deviation is lower for V3, which indicates a reliable convergence behavior over time. Our approach is leading to superior results, since the amount of hardware realizable gates is very small which in return leads to compute graphs which are manageable. Otherwise, the combinatorial space will explode and our method will not lead to a gain in reasonable time.

4.2. Impact and scalability

The complexity of a compute graph increases exponentially with the number of qubits, the available gates and hence the possible gate combinations. Thus the presented approach is not suitable for a naive upscaling. To still achieve circuit optimization for a larger number of qubits (e.g. beyond four or five), the general idea and observation is that, if only short circuit blocks are selected (e.g. of length between three to seven), usually many wires are not connected to these gates and they only cause an increase of the underlined dimensions. Thus, we reduce the non-needed wires and map the selected gate sequence to a subspace only containing required qubits. After the analysis in the smaller qubit space and potential block reduction, the resulting circuit is mapped back to the original size. Figure 7 visualizes the general idea. This approach allows us to deal with arbitrary sizes, in the latter experiments we show results with up to 15 qubits. Please also note that, in general, non-connected circuit blocks (e.g. with differently involved qubits) can be optimized in a parallel framework, which can speed up the optimization for complex architectures. We also observed that the compressing rate for larger qubit numbers decreases as the likelihood for non-reducible circuit blocks increases. Thus, our algorithm is best suited for very long circuits with a smaller amount of qubits as then more algebraic manipulations are necessary to reduce the circuit length. These more complex manipulations are not well covered by the baseline optimizers which explains our superior performance in the latter experiments.

Our approach is applicable to arbitrary (discrete) gate sets. To illustrate this we now present optimized circuit maps for an ion-trap architecture consisting of RX, RY, RZ gates and RXX gates as well as for NISQ-architectures as provided by IBM, consisting of RX, RZ and CZ gates. The proposed method is compared to a qiskit transpiler as well as the BQSKit compiler [68] on different optimization levels. The description in qiskit only states vague information, e.g. for its highest level: *level 3 pass manager: significant optimization by noise adaptive qubit mapping and gate cancellation using commutativity rules and unitary synthesis* [39]. The BQSKit is a very powerful software, with gate deletion and rule-free gate transpilation algorithms. Several well known algorithms (leap, qsearch, qfast, qpredict, PAS and more) are released as part of the BQSKit code [10, 13, 54, 65, 67]. As BQSKit outperforms several commercial compilers (Qiskit, Cirq, Tket), we use it as additional baseline for our experiments.

Tables 3–5 summarize results for random example circuits using different numbers of qubits and show that our proposed method is superior compared to all optimization levels. In all tables, the symbol ↓ indicates the objective to minimize the respective gate count. It should be noted that the qiskit transpilation works very efficiently and takes under a second, whereas the BQSKit and our optimizer take much longer (up to several minutes) to terminate (see table 2).

In the final experiment we evaluate the overall reduction which can be achieved, in comparison to the baselines BQSKit and qiskit. We generated 100 random quantum circuits of length 300 and evaluate

Table 3. Quantum circuit optimization example for an ion-trap architecture and comparison of our method to the qiskit optimizer on levels 1–3 and the BQSKit compiler on levels 2–4 (four qubits).

Method	# *RX* ↓	# *RY* ↓	# *RZ* ↓	# *RXX* ↓
original	82	71	86	61
Q-L1	36	43	63	61
Q-L2	41	48	58	60
Q-L3	41	48	58	60
B-L2	59	10	65	68
B-L3	54	9	72	59
B-L4	69	0	79	58
Ours	**9**	**27**	**38**	**38**

Table 4. Quantum circuit optimization example for a nisq architecture (IBM) and comparison of our method to the qiskit optimizer on levels 1–3 and the BQSKit compiler on levels 2–4. (6 qubits).

Method	# *RX* ↓	# *RZ* ↓	# *CZ* ↓
original	93	100	107
Q-L1	64	66	93
Q-L2	63	39	66
Q-L3	63	39	66
B-L2	82	100	94
B-L3	84	111	68
B-L4	90	132	60
Ours	**51**	**22**	**60**

Table 5. Quantum circuit optimization example for a nisq architecture (IBM) and comparison of our method to the qiskit optimizer on levels 1–3 and the BQSKit compiler on optimization levels 2–4. (15 qubits).

Method	# *RX* ↓	# *RZ* ↓	# *CZ* ↓
original	96	91	313
Q-L1	74	74	285
Q-L2	74	36	203
Q-L3	74	36	203
B-L2	87	112	306
B-L3	126	164	269
B-L4	293	395	253
Ours	**72**	**26**	**191**

Table 6. Statistics for quantum circuit optimization for an ion trap architecture and comparison of our method to the qiskit optimizer on levels 1–3 and the BQSKit compiler on optimization levels 2–4 over 100 runs (4 qubits). The values are the mean and in brackets we provide the standard deviation.

Method	# *RX* ↓	# *RY* ↓	# *RZ* ↓	# *RXX* ↓
In	78 (± 7)	83 (± 8)	78 (± 7)	59 (± 7)
Q-L1	32 (± 5)	46 (± 4)	59 (± 5)	59 (± 7)
Q-L2	33 (± 6)	49 (± 4)	66 (± 9)	56 (± 8)
Q-L3	33 (± 6)	49 (± 5)	66 (± 9)	56 (± 8)
B-L2	49 (± 10)	2 (± 2)	66 (± 15)	37 (± 6)
B-L3	39 (± 8)	1 (± 1)	57 (± 11)	32 (± 7)
B-L4	40 (± 7)	0 (± 0)	58 (± 19)	28 (± 4)
Ours	10 (± 3)	29 (± 6)	29 (± 5)	43 (± 8)

the mean and standard deviation of the reduced circuits, for both a nisq architecture and an ion trap architecture.

Table 6 and figure 8 summarize the statistics for quantum circuit optimization for an ion trap architecture and a comparison of our method to the qiskit optimizer on levels 1–3 and the BQSKit compiler

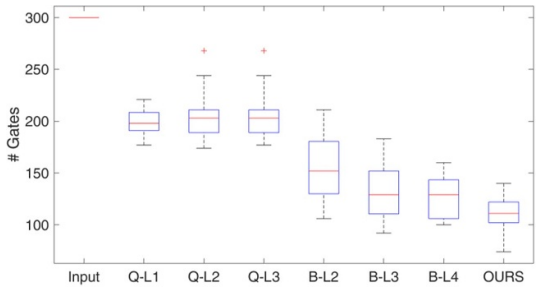

Figure 8. Statistical summary of the baseline compilation methods qiskit and BQSKit, compared to our proposed method. 100 different randomly sampled quantum circuits of length 300 (4 qubits) are reduced using the optimization levels 1–3 in qiskit (denoted as Q-L1 . . . Q-L3) and the optimization levels 2–4 in BQSKit (denoted as B-L2 . . . B-L4). Our results are denoted as OURS. The *y*-axis shows a box-plot. As the Input is of size 300, there is no standard deviation in this column. The gate set consists of *RX*, *RY*, *RZ* and *RXX* gates, as example for an ion trap architecture.

Table 7. Statistics for quantum circuit optimization for a nisq architecture (IBM) and comparison of our method to the qiskit optimizer on levels 1–3 and the BQSKit compiler on optimization levels 2–4 over 100 runs (4 qubits). The values are the mean and in brackets we provide the standard deviation.

Method	# *RX* ↓	# *RZ* ↓	# *CZ* ↓
In	108 (± 9)	109 (± 8)	82 (± 8)
Q-L1	59 (± 5)	68 (± 5)	69 (± 7)
Q-L2	59 (± 6)	39 (± 4)	51 (± 6)
Q-L3	59 (± 5)	39 (± 4)	51 (± 6)
B-L2	67 (± 7)	85 (± 11)	62 (± 8)
B-L3	55 (± 10)	70 (± 14)	39 (± 8)
B-L4	56 (± 10)	75 (± 15)	37 (± 7)
Ours	45 (± 6)	19 (± 4)	43 (± 6)

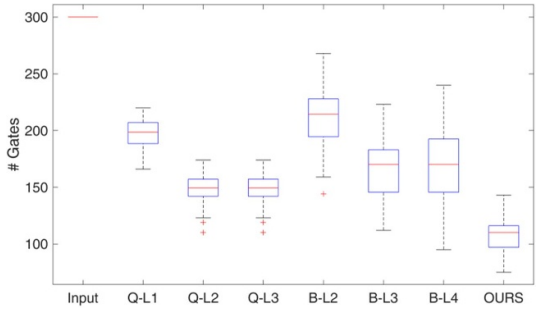

Figure 9. Statistical summary of the baseline compilation methods qiskit and BQSKit, compared to our proposed method. The experiment is similar to figure 8 using 100 different randomly sampled quantum circuits of length 300 (4 qubits). The gate set for reduction consists of *RX*, *RZ* and *CZ* gates.

on optimization levels 2–4 over 100 runs. The same is shown in Table 7 and figure 9 for a NISQ architecture (IBM). We observed severe performance differences between qiskit and BKSKit across the used gate sets. In contrast, our method is superior, and independent of the available gate set.

4.3. Quantum hardware experiments

In the next experiment we demonstrate that these optimizations are significant, when brought to an existing quantum chip. We therefore compare the measurement outcomes for equivalent circuits (in terms of the resulting unitary) with different transpilation lengths on two different chips. For this, we generated a random two qubit quantum circuit of length 40 (long) and reduced it to a quantum length of 8 (short), keeping the implemented unitary intact. Thus both quantum circuits are equivalent, but we expect a higher noise ratio for the more inefficient quantum circuit. Both quantum circuits have been transpiled in qiskit [1, 21] and evaluated in the IBM quantum platform [40]. Two different quantum chips have been used, both based on the IBM Eagle r3 architecture. This chip has 127 qubits, 5 K CLOPS and they are named IBM: Brisbane and IBM: Kyiv on the platform. Whereas the IBM: Brisbane runs on

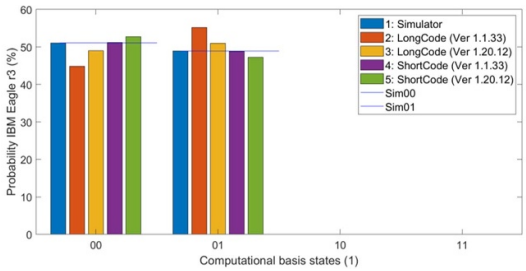

Figure 10. Measurement outcomes on two quantum chips (IBM) compared to the simulation result (most left). We provide two equivalent quantum circuits, one is named *long* for the sequence length of 40 and the other (equivalent on) is named *short* with a sequence length of 8. Both circuits are equivalent and the deviation from the simulation outcome (in blue) shows the importance to compile short circuits as decoherence over longer gate sequences is increasing significantly.

version 1.1.33 with an EPLG (error per layered gate for a 100 qubit chain) of 4.17%, the IBM:Kyiv processor runs on version 1.20.12 and has an EPLG of only 1.7% (according to the online documentation [41]). Figure 10 summarizes the obtained measurements. Figure 10 shows the simulation (in blue) as left most bar and the different circuits on the two used processors in the other bars. The measurements in figure 10 show that the more recent version 1.20.12 provides a more stable outcome, compared to version 1.1.33, which is in accordance to the provided EPLG scores. While this experiment is based on a two-qubit circuit and a length of 40 gates which is nearly trivial, decoherence is still apparent and efficient transpiled quantum circuits are mandatory, now and in the future.

To summarize the experiments, our reduced quantum circuits are transpiled to significantly more efficient hardware implementable circuits and they lead to better results which are closer to the expectation provided by the simulator.

5. Summary and discussion

Naively mapping a quantum circuit to an existing hardware can lead to a long quantum circuit with unnecessary redundancies. Due to decoherence in quantum computers, it is mandatory to ensure we find equivalent, shorter and more efficient, circuits. In this work we presented three different variants, based on a local term replacement scheme, to substantially reduce circuit length while maintaining the unitary implemented by the circuit. The first variant is based on a stochastic search scheme, the other variants are driven by a DR scheme and a ML based decision support. We show that quantum circuit length can be efficiently reduced and different modifications can be done to significantly boost the compute time for optimization of the circuit length. It should be noted that our method is significantly slower than the qiskit transpilers and our approach is only useful when a highly efficient circuit block is required and reused several times in a larger context. It is also possible to perform simple reductions with the available transpilers and then to refine them further with our method. Thus, they are not exclusive to each other. We also performed experiments using the *ZX*-calculus [60], but as discussed in other works, such as [48], the outcome can be less efficient as the original input circuit. As we observed the same using *ZX*-calculus for our gate sets, there is need for further investigations and optimization, e.g. based on reinforcement learning. This will be part of future research. We similarly experimented with the recently published IBM-AI model [25] for transpilation. It turned out that the transpiler does not work with an arbitrary set of gates and it returned the level-3 transpiled code. We will address this as part of future research, as well. In future works, we will also integrate costs for decoherence time of single operators and operator chains. E.g. a token chain with five operators and two *CZ*-gates might have a higher decoherence than a seven operator block with only one *CZ* gate. With such cost measures we will replace our optimization criteria (which is just the number of gates at the moment) with alternatives in the future. To increase the sampling efficiency, we will also explore methods based on neural guided sampling, as proposed in [24].

Data availability statement

No new data were created or analyzed in this study.

Acknowledgments

This work was supported, in part, by the Federal Ministry of Education and Research (BMBF), Germany under the AI service center KISSKI (Grant No. 01IS22093C), by the Quantum Valley Lower Saxony, by Germany's Excellence Strategies EXC-2122 PhoenixD and EXC-2123 Quantum Frontiers and the DFG via the project ResourceQ. The authors would also like to thank the Qudora team and Ed Younis (Berkley) for their valuable feedback. We also thank Ritajit Majumdar (IBM Quantum) for the fruitful discussions and insights on the IBM transpiler toolset. We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

ORCID iDs

Bodo Rosenhahn 0000-0003-3861-1424
Tobias J Osborne 0000-0002-0688-2345
Christoph Hirche 0000-0001-9265-827X

References

[1] Asfaw A *et al* 2020 *Learn quantum computation using qiskit* (available at: https://github.com/Qiskit/textbook/tree/main/notebooks/ch-applications#)
[2] Ashhab S, Yoshihara F, Tsuji M, Sato M and Semba K 2024 Quantum circuit synthesis via a random combinatorial search *Phys. Rev. A* **109** 052605
[3] Baker B, Gupta O, Raskar R and Naik N 2018 Accelerating neural architecture search using performance prediction *6th Int. Conf. on Learning Representations, ICLR 2018, (Vancouver, BC, Canada, 30 April–3 May 2018), Workshop Track Proc.* (OpenReview.net)
[4] Blum C and Roli A 2003 Metaheuristics in combinatorial optimization: overview and conceptual comparison *ACM Comput. Surv.* **35** 268–308
[5] Brandhofer S, Polian I and Büchler H P 2021 Optimal mapping for near-term quantum architectures based on rydberg atoms *2021 IEEE/ACM Int. Conf. On Computer Aided Design (ICCAD)* pp 1–7
[6] Breiman L 2001 Random forests *Mach. Learn.* **45** 5–32
[7] Breiman L, Friedman J, Stone C J and Olshen R A 1984 *Classification and Regression Trees* (Chapman and Hall/CRC)
[8] Cai H, Chen T, Zhang W, Jun Y and Wang Y 2018 Efficient architecture search by network transformation *Proc. 32nd AAAI Conf. on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18) and the 8th AAAI Symp. on Educational Advances in Artificial Intelligence (EAAI-18), (New Orleans, Louisiana, USA), 2018* (AAAI Press) pp 2787–94
[9] Chin T-J, Purkait P, Eriksson A and Suter D 2015 Efficient globally optimal consensus maximisation with tree search *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*
[10] Cincio L, Rudinger K, Sarovar M and Coles P J 2021 Machine learning of noise-resilient quantum circuits *PRX Quantum* **2** 010324
[11] Datta K, Kole A, Sengupta I and Drechsler R 2022 Mapping quantum circuits to 2-dimensional quantum architectures *INFORMATIK* vol 2022
[12] Datta K, Kole A, Sengupta I and Drechsler R 2022 Nearest neighbor mapping of quantum circuits to two-dimensional hexagonal qubit architecture *2022 IEEE 52nd Int. Symp. on Multiple-Valued Logic (ISMVL)* pp 35–42
[13] 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)* pp 223–34
[14] de Moura L and Bjørner N 2008 Z3: an efficient smt solver *Tools and Algorithms for the Construction and Analysis of Systems* ed C R Ramakrishnan and J Rehof (Springer) pp 337–40
[15] Ding Z, Ko T, Yao J, Lin L and Li. X 2024 Random coordinate descent: a simple alternative for optimizing parameterized quantum circuits *Phys. Rev. Res.* **6** 033029
[16] Franken L, Georgiev B, Mucke S, Wolter M, Heese R, Bauckhage C and Piatkowski N 2022 Quantum circuit evolution on NISQ devices *2022 IEEE Congress on Evolutionary Computation (CEC)* pp 1–8
[17] George E and McCulloch R 1993 Variable selection via Gibbs sampling *J. Am. Stat. Assoc.* **88** 881–9
[18] Gilks W R and Wild P 1992 Adaptive rejection sampling for Gibbs sampling *J. R. Stat. Soc. C* **41** 337–48
[19] He Z, Deng M, Zheng S, Li L and Situ H 2024 Training-free quantum architecture search *Proc. AAAI Conf. on Artificial Intelligence* vol 38 pp 12430–8
[20] Hopcroft J E and Ullman J D 1979 *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley Publishing Company)
[21] Javadi-Abhari A *et al* 2024 Quantum computing with Qiskit
[22] Kaye P, Laflamme R and Mosca M 2007 *An Introduction to Quantum Computing* (Oxford University Press, Inc.)
[23] Kjellström H, Kragić D and Black M J 2010 Tracking people interacting with objects *2010 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* pp 747–54
[24] Kluger F, Brachmann E, Ackermann H, Rother C, Yang M Y and Rosenhahn B 2020 Consac: robust multi-model fitting by conditional sample consensus *Computer Vision and Pattern Recognition (CVPR)*
[25] Kremer D, Villar V, Paik H, Duran I, Faro I and Cruz-Benito J 2024 Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning (arXiv:2405.13196)
[26] Heras U L, Alvarez-Rodriguez U, Solano E and Sanz M 2016 Genetic algorithms for digital quantum simulations *Phys. Rev. Lett.* **116** 230504
[27] Li Li, Fan M, Coram M, Riley P and Leichenauer S 2020 Quantum optimization with a novel Gibbs objective function and ansatz architecture search *Phys. Rev. Res.* **2** 023074

[28] Li Y, Liu W and Li. M 2024 Deep reinforcement learning for mapping quantum circuits to 2d nearest-neighbor architectures *Adv. Quantum Technol.* **7** 2300289
[29] Marcus M P 1967 *Switching Circuits for Engineers* (Prentice-Hall)
[30] Martyniuk D, Jung J and Paschke A 2024 Quantum architecture search: a survey (arXiv:2406.06210)
[31] Mathesen L, Pedrielli G, Ng S H and Zabinsky Z B 2021 Stochastic optimization with adaptive restart: a framework for integrated local and global learning *J. Glob. Optim.* **79** 87–110
[32] Meng Q and Ng. S H 2016 Combined global and local method for stochastic simulation optimization with an aglgp model *2016 Winter Simulation Conf. (WSC)* pp 827–38
[33] Metropolis N and Ulam. S 1949 The Monte Carlo method *J. Am. Stat. Assoc.* **44** 335–41
[34] Miikkulainen R 2020 *Neuroevolution* (Springer) pp 1–8
[35] Murali P, Debroy D M, Brown K R and Martonosi M 2020 Architecting noisy intermediate-scale trapped ion quantum computers *2020 ACM/IEEE 47th Annual Int. Symp. on Computer Architecture (ISCA)* pp 529–42
[36] Negrinho R, Patil D, Le N, Ferreira D, Gormley M and Gordon G 2019 Towards modular and programmable architecture search *Advances in Neural Information Processing Systems*
[37] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information* (Cambridge University Press)
[38] Rosenhahn B, Osborne T J and Hirche C 2025 (available at: www.tnt.uni-hannover.de/en/staff/rosenhahn/QCOptimDemo.zip)
[39] IBM Quantum 2024 (available at: https://github.com/Qiskit/qiskit/blob/main/qiskit/transpiler/preset_passmanagers/level3.py) (Accessed August 2024)
[40] IBM Quantum 2024 (available at: https://quantum.ibm.com) (Accessed August 2024)
[41] IBM Quantum 2024 (available at: https://quantum.ibm.com/services/resources?type = Eagle) (Accessed August 2024)
[42] Ovide A, Santiago R, Bandic M, van Someren H, Feld S, Abadal S, Alarcon E and Almudever C 2023 Mapping quantum algorithms to multi-core quantum computing architectures *IEEE Int. Symp. on Circuits and Systems (ISCAS)* pp 1–5
[43] Pardalos P M and Romeijn H E 2002 *Handbook of Global Optimization* vol 2 (Kluwer Academic Publishers)
[44] Pereyra M, Schniter P, Chouzenoux Emilie, Pesquet J-C, Tourneret J-Y, Hero A O and McLaughlin S 2016 A survey of stochastic simulation and optimization methods in signal processing *IEEE J. Sel. Top. Signal Process.* **10** 224–41
[45] Powell W and Lyu H 2024 Stochastic optimization with arbitrary recurrent data sampling *Proc. 41st Int. Conf. on Machine Learning* (*ICML'24*) (JMLR.org)
[46] Quinlan R 1986 Induction of decision trees *Mach. Learn.* **1** 81–106
[47] Rasconi R and Oddi A 2019 An innovative genetic algorithm for the quantum circuit compilation problem *Proc. AAAI Conf. on Artificial Intelligence* vol 33 pp 7707–14
[48] Riu J, Nogué J, Vilaplana G, Garcia-Saez A and Estarellas M P 2024 Reinforcement learning based quantum circuit optimization via zx-calculus
[49] Rosenhahn B 2023 Optimization of sparsity-constrained neural networks as a mixed integer linear program *J. Optim. Theory Appl.* **199** 931–54 open access
[50] Rosenhahn B and Hirche C 2024 Quantum normalizing flows for anomaly detection *Phys. Rev. A* **110** 022443
[51] Rosenhahn B and Tobias J O 2023 Monte Carlo graph search for quantum circuit optimization *Phys. Rev. A* **108** 062615
[52] Ryabov V A 2015 Quantum volume *Int. J. Mod. Phys. B* **29** 1550166
[53] Schoen F 2002 Two-phase methods for global optimization *Handbook of Global Optimization, Volume 2*, ed P M Pardalos and H E Romeijn (Kluwer Academic Publishers) pp 151–77
[54] Schönberger M, Scherzinger S and Mauerer W 2023 Ready to leap (by co-design)? Join order optimisation on quantum hardware *Proc. ACM Manag. Data* **1** 1–27
[55] Solis F J and Wets R J-B 1981 Minimization by random search techniques *Math. Oper. Res.* **6** 19–30
[56] Spall J C 2003 *Introduction to Stochastic Search and Optimization* (*Wiley Series in Discrete Mathematics and Optimization*) (Wiley)
[57] De Stefano M D, Di Nucci D D, Palomba F and De Lucia A D 2024 An empirical study into the effects of transpilation on quantum circuit smells *Empir. Softw. Eng.* **29** 61
[58] Tang Z, Hu X and Périaux J 2020 Multi-level hybridized optimization methods coupling local search deterministic and global search evolutionary algorithms *Arch. Comput. Methods Eng.* **27** 939–75
[59] R O Topaloglu ed 2023 *Design Automation of Quantum Computers* vol 2023 (Springer)
[60] van de Wetering J 2020 Zx-calculus for the working quantum computer scientist (arXiv:2012.13966)
[61] J van Leeuwen ed 1991 *Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics*, (MIT Press)
[62] Wang P, Usman M, Parampalli U, Hollenberg L C L and Myers C R 2023 Automated quantum circuit design with nested Monte Carlo tree search *IEEE Trans. Quantum Eng.* **4** 1–25
[63] Watabe M, Shiba K, Chen C-C, Sogabe M, Sakamoto K and Sogabe T 2021 Quantum circuit learning with error backpropagation algorithm and experimental implementation *Quantum Rep.* **3** 333–49
[64] Wauters M M, Panizon E, Mbeng G B and Santoro G E 2020 Reinforcement-learning-assisted quantum optimization *Phys. Rev. Res.* **2** 033446
[65] Wu X-C, Davis M G, Chong F T and Iancu C 2021 Reoptimization of quantum circuits via hierarchical synthesis *2021 Int. Conf. on Rebooting Computing (ICRC)* pp 35–46
[66] Xie S, Zheng H, Liu C and Lin L 2019 SNAS: stochastic neural architecture search *Int. Conf. on Learning Representations*
[67] Younis E and Iancu C 2022 Quantum circuit optimization and transpilation via parameterized circuit instantiation (arXiv:2206.07885)
[68] Younis E, Iancu C C, Lavrijsen W, Davis M and Smith E (USDOE) 2021 Berkeley quantum synthesis toolkit (bqskit) v1 (available at: https://bqskit.lbl.gov)
[69] Zabinsky Z B 1998 Stochastic methods for practical global optimization *J. Glob. Optim.* **13** 433–44
[70] Zhang S-X, Hsieh C-Y, Zhang S and Yao H 2021 Neural predictor based quantum architecture search *Mach. Learn.: Sci. Technol.* **2** 045027
[71] Zhang S-X, Hsieh C-Y, Zhang S and Yao H 2022 Differentiable quantum architecture search *Quantum Sci. Technol.* **7** 045023
[72] Zhou L, Wang S-T, Choi S, Pichler H and Lukin M D 2020 Quantum approximate optimization algorithm: performance, mechanism and implementation on near-term devices *Phys. Rev. X* **10** 021067
[73] Zhu W, Pi J and Peng Q 2023 A brief survey of quantum architecture search *Proc. 6th Int. Conf. on Algorithms, Computing and Systems* (*ICACS '22*) (Association for Computing Machinery)
[74] Zulehner A, Paler A and Wille R 2018 An efficient methodology for mapping quantum circuits to the IBM QX architectures *IEEE Trans. Comput. Aided Design Integr. Circ. Syst.* **38** 1226–36