• REVIEW •

Quantum Circuit Synthesis and Compilation Optimization: Overview and Prospects

Ge Yan[1,2], Wenjie Wu[2,4], Yuheng Chen[1,2], Kaisen Pan[1,2], Xudong Lu[1,2], Zixiang Zhou[1,2], Yuhan Wang[2,5], Ruocheng Wang[2,5] & Junchi Yan[1,2,3*]

[1]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China;*
[2]*MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai 200240, China;*
[3]*School of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai 200030, China;*
[4]*Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China;*
[5]*Zhiyuan College, Shanghai Jiao Tong University, Shanghai 200240, China*

Abstract   Quantum computing is regarded as a promising paradigm that may overcome the current computational power bottlenecks in the post-Moore era. The increasing maturity of quantum processors, especially superconducting ones, provides more possibilities for the development and implementation of quantum algorithms. As the crucial stages for quantum algorithm implementation, the logic circuit design and quantum compiling have also received significant attention, which covers key technologies such as quantum logic circuit synthesis (also widely known as quantum architecture search) and optimization, as well as qubit mapping and routing. Recent studies suggest that the scale and precision of related algorithms are steadily increasing, especially with the integration of artificial intelligence methods. In this survey, we systematically review and summarize a vast body of literature, exploring the feasibility of an integrated design and optimization scheme that spans from the algorithmic level to quantum hardware, combining the steps of logic circuit design and compilation optimization. Leveraging the exceptional cognitive and learning capabilities of AI algorithms, one can reduce manual design costs, enhance the precision and efficiency of execution, and facilitate the implementation and validation of the superiority of quantum algorithms on hardware.

Keywords   quantum circuit synthesis, quantum architecture search, quantum circuit optimization, quantum compiling, quantum machine learning, artificial intelligence

1 Introduction

In recent years, quantum computing and quantum machine learning have garnered extensive attention. Due to their unique physical properties, quantum computing is considered the most promising new computing architecture to break through the existing computational power bottlenecks in the post-Moore era. Several notable quantum algorithms, such as Shor's algorithm for large integer factorization [170] and Grover's algorithm for unstructured search [64], have been theoretically proven to enjoy quantum superiority over classical algorithms. With the advent of the Noisy Intermediate-Scale Quantum (NISQ) era in the past decade [80, 157], quantum superiority has not only been demonstrated theoretically but also experimentally verified on quantum prototypes. For example, Google has demonstrated quantum superiority on a superconducting quantum[1] prototype through random circuit sampling tasks [12, 197]. Similarly, the team with USTC has verified this through random boson sampling tasks on a photonic quantum prototype [211].

Compared to random circuit sampling and random boson sampling, exploring the superiority of non-trivial and practically significant quantum algorithms presents a significant challenge, particularly given the constraints of existing quantum hardware, in terms of e.g., coherence time, noise, and physical topology. Many existing quantum algorithms utilize oracles [64, 170, 189], which are black-box unitary matrices where the corresponding unitary operation is known yet whose implementations are not explicitly

---

* Corresponding author (email: yanjunchi@sjtu.edu.cn)

   1) For conciseness and clarity, when referring to quantum hardware in this article, the default is the superconducting system, which is a mainstream quantum hardware technology.

Figure 1 Quantum algorithm implementation pipeline. A quantum algorithm can be written in the form of multiple unitary transformations. logic circuit synthesis and optimization methods are then applied to obtain logic circuits. We utilize qubit mapping and routing methods to build an executable program during the quantum compiling stage. In this paper, we mainly focus on logic circuit design and quantum compiling.

defined. To best execute the algorithms, one needs to implement these unitary operators in the algorithms with minimal cost (such as circuit depth, gate count, etc.) and thus obtain the corresponding logic circuits of the algorithms. These logic circuits are further optimized with specific optimization goals. For instance, in the NISQ era, the focus is on optimizing two-qubit gates, while in the fault-tolerant quantum computing era, the emphasis shifts to optimizing T gates. Finally, the logic circuits must be compiled, requiring the optimal mapping from logical qubits to physical qubits and the addition of necessary SWAP gates to satisfy the circuit's connectivity requirements. For variational quantum algorithms, we can also design and optimize parameterized quantum circuits, breaking traditional design paradigms. By integrating hardware constraints, we seek a balance between theoretical precision and noise (where reducing circuit depth decreases theoretical precision, but reducing noise can decrease execution errors) to achieve optimal results.

Figure 1 shows the entire process from quantum algorithm design to execution. This process includes transforming a quantum algorithm into unitary transformations, generating logic circuits through synthesis and optimization methods, and then compiling these circuits, considering the physical qubit topology and other quantum hardware constraints, into executable quantum programs on target quantum processors. In other words, the transformation from a quantum algorithm to an executable quantum program can be divided into three steps: quantum algorithm design, quantum logic circuit design, and quantum compiling. In reviews of modern quantum computing technology [81], these steps are often collectively referred to as quantum circuit compilation, with relevant research briefly summarized. Early work also roughly summarized quantum circuit compilation in the NISQ era with heuristic methods [102], including genetic algorithms, genetic programming, ant colony optimization, and planning algorithms. In this paper, we focus on efficiently transforming quantum algorithms into executable quantum programs by comprehensively summarizing research on logic circuit synthesis and optimization, as well as qubit mapping and routing. Additionally, we provide prospects and insights into automatic quantum logic circuit design and compilation optimization, which enables researchers to focus more on algorithm design, fully leveraging the potential of NISQ-era quantum hardware and promoting the practical application and superiority verification of quantum algorithms.

This survey first introduces and discusses existing major methods for representing quantum circuits. By representing circuits in various forms, such as gate model [46], directed acyclic graph [123], phase polynomial [7], ZX diagram [36], and tensor network [22], we can explore different synthesis, optimization, and compilation methods corresponding to these representation forms. For the problem of quantum logic circuit synthesis, we first examine the applications, which are divided into cases with and without target unitary matrices. Applications with given unitary matrices include oracle implementation [23,158], while applications without target unitary matrices encompass variational quantum algorithms [48, 63, 195], error correction codes [33,37], and nonlinear unit design [159]. Furthermore, we explore the methods for

quantum circuit synthesis. We can generally classify the algorithms by dividing the task into two steps — deciding circuit structure and evolving rotation gate parameters (if applicable). The introduction of machine learning algorithms has made it possible to search for quantum circuits containing parameters, yielding better results than traditional methods.

For the problem of quantum logic circuit optimization, we have categorized relevant literature from the perspectives of optimization objectives and methods. As an important step following circuit synthesis, quantum circuit optimization aims to obtain a better equivalent circuit based on the given optimization objectives. Considering the differing characteristics of the NISQ era and the fault-tolerant quantum computing era, related studies have proposed various optimization objectives, which are also tied to the quantum circuit representation method. At the algorithm level, besides borrowing methods such as pattern matching and peephole optimization from traditional compilation optimization [127], we can also reduce the circuit optimization problem to other well-studied problems [9, 75, 166] or build an equivalent circuit database based on a specific set of quantum gates, obtaining the optimal subcircuit through searching the database [8, 26, 99]. Additionally, reinforcement learning methods have also recently achieved notable success in the field of quantum circuit optimization [56, 110, 161], and can even self-discover new matching patterns [163], surpassing traditional methods. We also summarize optimization methods for variational quantum algorithms. These studies, based on specific variational quantum algorithms [63, 117, 177, 204], propose corresponding optimization methods to make the optimized variational quantum circuits more suitable for quantum processors in the NISQ era.

In the quantum circuit compiling state, we need to consider the topological structure of the physical qubits, specifically the connectivity between them. Two-qubit quantum gates cannot be applied between qubits that lack a direct coupler, so some gates in the logic circuit obtained during the logical circuit design stage cannot be directly executed on a quantum computer. By adding SWAP gates to the circuit, one can exchange the information and state of two qubits; however, each SWAP gate requires three CNOT gates for implementation. Introducing too many SWAP gates into the circuit can severely impact operational accuracy. Therefore, we sorted out the algorithms for quantum qubit mapping and routing. Similar to logical circuit optimization, qubit mapping and routing focus on the number of quantum gates and circuit depth, but they also emphasize runtime and fidelity [113, 114, 135, 144, 145, 178]. Due to the optimization nature of the algorithm, qubit routing methods are akin to logical circuit optimization methods, including exact solutions [172], heuristic methods [217], reduction [130, 135, 191], and machine learning approaches [77, 155, 171].

This leads to a key question explored in this paper: is it possible to develop an integrated algorithm that encompasses both quantum logical circuit design and quantum compiling? Such an algorithm should include the following steps: (1) Generate the optimal logical circuit corresponding to the quantum algorithm based on the built-in gate set of the quantum processor; (2) Complete the selection, mapping, and routing of quantum bits and couplers according to the calibration data of the quantum processor; (3) Balance theoretical accuracy and practical noise, achieving the optimal performance of the quantum processor with an appropriate trade-off in theoretical accuracy. Solving all these constraints is a challenging problem. However, the introduction of artificial intelligence may facilitate the design of related algorithms. Relevant literature indicates that with the introduction of machine learning algorithms, quantum logical circuit synthesis, optimization, and compilation have become significant application scenarios for AI4Science. Ideally, integrating all the above steps into the compiler of the quantum hardware cloud platform could automate these processes through AI algorithms. This would significantly enhance the practical performance of the submitted algorithms, reducing the additional labor costs post-quantum algorithm design, and providing a new paradigm for quantum computing applications.

In summary, the contributions of this paper are as follows: (1) Conducting a comprehensive analysis of the various steps and challenges in the development and execution of quantum algorithms, covering the entire process from quantum logical circuit design to quantum compiling; (2) Systematically organizing the algorithms and their characteristics related to quantum logical circuit synthesis, optimization, as well as qubit mapping and routing; (3) Analyzing the requirements for real-world implementation of quantum algorithms in the NISQ era, and proposing an integrated quantum circuit design and compiling solution. This solution leverages artificial intelligence to enhance the performance and design efficiency of quantum algorithms and envisions a new paradigm for quantum compiling in the age of artificial intelligence.

Figure 2 Quantum Circuit Representation. Gate Model: most common representation method; DAG: example DAG transformed from the gate model example, "start" and "end" nodes are added to complete the graph; QMDD: QMDD representation of a three-qubit quantum operation, we present the transformation matrix and the layerwise structure as well as the size of the corresponding matrix of each node; Tensor Network: a quantum state $|\psi\rangle$ can be represented by multiple common types of tensor network; ZX-Diagram: the green and red spider in ZX-diagram as well as an example ZX-diagram transformed from gate model example.

2   Quantum Circuit Representation

We begin by discussing the representation of quantum circuits, which involves using graphs or vectors to depict specific information about the circuits. The representation of quantum circuits is crucial for their synthesis and optimization. Some synthesis and optimization algorithms are based on specific representations. Different representations of the same circuit can reduce the complexity of subsequent circuit design and compiling steps while still conveying the necessary information.

2.1   Quantum Gate Model

Currently, the most common and fundamental method for representing quantum circuits is through the quantum gate model. This method characterizes quantum circuits using a system similar to classical gate circuits by representing the operations on qubits as quantum gates. A quantum gate can operate on multiple qubits. Each gate corresponds to a unitary transformation, modifying the quantum state and enabling circuit operations. Quantum gates can be divided into two main categories: parameterized and non-parameterized gates. The former allows for continuous transformations. To implement quantum algorithms, a set of quantum gates that can approximate any unitary matrix with arbitrary precision is required. This set is known as the universal quantum gate set [142]. According to [142], the necessary and sufficient condition for a universal quantum gate set is that it can approximate any single-qubit unitary transformation and includes a two-qubit gate. Other two or more qubit gates can be constructed from this universal set.

One of the most commonly used and well-studied universal quantum gate sets is Clifford+T [55]. The Clifford set includes H gates, S gates, and CNOT gates [60,62], and quantum circuits containing only Clifford gates can be efficiently simulated by classical computers, making this set particularly important [61].

The Clifford set forms the foundation for many quantum algorithms and is defined as follows:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \sqrt{Z}, \quad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1}$$

Through the combination of these three gates, we can generate a variety of other gates. For example, the Pauli-Z gate can be obtained by S^2, the Pauli-X gate can be obtained by $HZH^\dagger$, and the Pauli-Y gate can be obtained by $SXS^\dagger$. However, these basic Clifford gates are not universal on their own. They cannot approximate many other gates with arbitrary precision within a finite sequence. To achieve universal quantum computation, we need to supplement the Clifford set with the T gate:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = \sqrt{S} = \sqrt[4]{Z}. \tag{2}$$

The T gate is a phase gate. When we supplement the Clifford set with the non-Clifford T gate, the quantum gate set can continuously generate new non-Clifford gates, making the H, S, T, CNOT set capable of approximating any quantum gate. The Clifford+T set is the most commonly used universal quantum gate set, primarily because Clifford gates are relatively easy to implement on superconducting quantum computers. However, the T gates are more challenging to implement, so one major objective of quantum circuit optimization is to minimize the number of T gates required.

Besides the Clifford+T set, there are other general quantum gate sets, such as the parameterized rotation gates Rx(θ), Ry(θ), and Rz(θ), along with the CNOT gate, which is commonly used in quantum machine learning. The unitary matrices for the three parameterized rotation gates are as follows:

$$Rx(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}, \quad Ry(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}, \quad Rz(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}}. \end{bmatrix} \tag{3}$$

Compared to the Clifford+T set, the main advantage of this universal gate set is that all single-qubit gates are parameterized, allowing for continuous transformations. As a result, the number of gates required to approximate arbitrary single-qubit gates is significantly reduced compared to using Clifford+T gates with discrete angles. However, the arbitrary angles of parameterized gates can affect accuracy when implemented on NISQ-era quantum computers. In the future fault-tolerant quantum computing era, it is also challenging to implement these gates at arbitrary angles through encoding.

Additionally, some universal quantum gate sets are designed for specific problems. For instance, the NCV quantum gate set is used to optimize circuits [122–124, 129], particularly multi-controlled-NOT gates. This set includes NOT gates, CNOT gates, and Controlled-V/V† gates, where the unitary matrix for the V gate is

$$V = \frac{1+i}{2} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}. \tag{4}$$

Some studies also use built-in gates on superconducting quantum computers directly, like the iSWAP= $e^{i\frac{\pi}{4}(XX+YY)}$ gate in [174]. These special cases will be explained in detail when encountered in the subsequent sections.

2.2   Directed Acyclic Graph

To better extract information from quantum circuits, we can use Directed Acyclic Graphs (DAGs) to represent quantum gate circuits and apply them to tasks such as quantum architecture search [70, 72] and quantum compiling [19, 107]. By adding a source node and a sink node at the beginning and end of the quantum circuit, respectively, and treating each quantum gate as a node, we create a directed graph. Adjacent gates on the same qubit are connected by directed edges, and multi-qubit gates introduce additional edges. These directed edges represent the sequence in which the gates are applied. The advantages of using DAGs to represent quantum circuits are: (1) the structure of quantum circuits naturally transforms into DAGs; (2) DAGs can represent the application order of quantum gates and the

control qubits of multi-qubit gates, maximizing the advantages of DAGs ; and (3) graph neural networks can effectively extract and compute features of nodes and edges, handling large-scale quantum circuits.

Additionally, there is a special circuit representation in the form of directed acyclic graphs called Quantum Multiple-Valued Decision Diagram (QMDD) [143]. As illustrated in Figure 2, starting from the q_2 level, which is a 2×2 matrix, we build the diagram layer by layer according to the transformation matrix. Each 2×2 matrix consists of four matrix elements. There are four types of q_2 level matrices in the circuit, so this level has four nodes. The q_1 level above comprises a 4×4 matrix. Each q_1 level matrix is composed of four q_2 level matrices, connecting each q_1 level node to four q_2 level nodes. The final q_0 level 8×8 unitary matrix is the target unitary matrix we aim to build. The completed QMDD diagram is shown at the bottom of the QMDD module in Figure 2.

2.3 Circuit Polynomial

Based on the representation of the quantum gate model, we can further characterize quantum circuits using algebraic methods, such as linear Boolean functions or phase polynomials [6, 8, 128, 136, 183]. These polynomial representation algorithms focus more on the quantum state rather than the specific quantum gates acting on the state. An arbitrary single-qubit quantum gate can be seen as inducing the following transformations on the quantum state:

$$|x\rangle \rightarrow e^{\mathrm{i}p(x)} |g(x)\rangle, \tag{5}$$

where $|x\rangle \rightarrow |g(x)\rangle$ is a reversible transformation, and $e^{\mathrm{i}p(x)}$ represents the phase information imparted by the quantum gate to the current quantum state $|x\rangle$. Since reversible transformations in quantum circuits are relatively straightforward to analyze, phase polynomial expressions focus more on the phase information contributed by quantum gates, particularly gates like Z, S, T, $\mathrm{Rz}(\theta)$ and CZ. Given that the T gate is a crucial target in quantum circuit optimization, phase polynomial expressions play a significant role in optimizing circuits. Below, we present a specific example of a phase polynomial. For an n-qubit circuit, assuming that there are only two types of gates, CNOT and $\mathrm{Rz}(\theta)$, the phase function $p(x)$ for the sample quantum circuit in Figure 2 can be expressed as:

$$p(x_1, x_2, x_3) = \theta_1(x_1) + \theta_2(x_1 \oplus x_2) + \theta_3(x_1 \oplus x_2 \oplus x_3) + \theta_4(x_2 \oplus x_3). \tag{6}$$

The quantum states before and after the execution of the transformation circuit can be expressed as:

$$|x_1, x_2, x_3\rangle \rightarrow e^{\mathrm{i}p(x_1, x_2, x_3)} |x_2 \oplus x_3, x_1 \oplus x_2 \oplus x_3, x_3\rangle. \tag{7}$$

Since the coefficients in the phase function are all rotation angles limited to a range of 2π, we can take the modulo of the superimposed angle. This allows the phase function to serve as a criterion for determining whether two circuits are equivalent. Compared to directly comparing two unitary matrices, using phase polynomials enables the equivalence assessment of larger-scale circuits. For instance, Ruiz et al. [163] optimized a 72-qubit quantum circuit based on phase polynomial, a task that cannot be accomplished by merely comparing unitary matrices.

2.4 Tensor Networks

As introduced by [151] for the mathematical concept of tensor networks, tensor can be regarded as a multidimensional vector, where scalars, vectors, and matrices can be considered as 0-dimensional, 1-dimensional, and 2-dimensional tensors, respectively [146]. Deutsch introduced the graphical representation of tensors into quantum computing, leading to the quantum gate circuits we discussed earlier. Therefore, quantum gate models are a special form of tensor networks. We will briefly introduce this more fundamental representation, with more detailed content about the tensor network available in [22]. Tensor networks can connect different tensors through indices, an operation called tensor network contraction. Some classic tensor network structures include Matrix Product State (MPS), Projected Entangled Pair States (PEPS), Tree Tensor Network (TTN), and Multi-Scale Entanglement Renormalization Ansatz (MERA), as shown in Figure 2.

Since matrix multiplication can be viewed as the contraction of two second-order tensors based on the same index, quantum circuits can naturally be simulated and optimized using tensor networks [120]. The main optimization method is to reduce the cost of contractions, but finding an optimal tensor

network contraction sequence is NP-hard [34]. This problem can be further modeled as finding an optimal tree decomposition of the tensor network graph structure [168]. The tree pruning method proposed in the paper can decompose the original graph into multiple parallel subgraphs with the minimum tree depth. Huang et al. [78] proposed index slicing, which divides the entire contraction into sub-tasks that can be executed in parallel. Specifically, a greedy algorithm selects indices that lead to the greatest reduction in complexity, accompanied by local reordering of the contraction tree. Additionally, pre-computing small tensors on the CPU and reordering matrix multiplication on the GPU can reduce the actual running cost. Contraction tasks can be divided into several tensor transpositions and one matrix multiplication [116]. Vincent et al. [185] constructed a task graph based on the dependencies of the aforementioned transpositions and multiplications, allowing different transpositions to be parallelized. Furthermore, by naming to ensure that shared operations between sliced contractions are only calculated once, costs are significantly reduced.

Another research direction is to optimize performance by simulating various tensor network structures when designing quantum circuits. These circuits are often referred to as Tensor Network Quantum Circuits (TNQC). Generative and adversarial tasks are completed using quantum circuits based on MPS and TTN [84]. Since TNQC allows us to simulate certain quantum circuits efficiently, we can use this method to save physical qubits and reduce the impact of real quantum computing noise. Quantum circuits based on MPS and MERA are also analyzed [67], finding that TNQC can use fewer quantum gate parameters to achieve the same accuracy, further validating the practicality of TNQC.

2.5   ZX Diagrams

ZX diagrams are a method for representing quantum circuits based on ZX calculus, proposed by Coecke and Duncan in 2008 [36]. ZX diagrams are also a special form of tensor networks that can represent any linear mapping between qubits [93]. They use spiders and the lines connecting spiders to represent a quantum circuit. Typically, ZX diagrams include two types of spiders: green Z spiders and red X spiders, as shown in Figure 2 [203]. When the angles are π, these two types of spiders correspond to the Pauli-Z and Pauli-X gates, respectively.

Spiders define the basic linear operations in any dimension, where the lines on the left side of a spider represent the inputs of the current linear operation, and the lines on the right side represent the outputs. Each spider's output can further connect to the inputs of other spiders, constructing a large network. ZX diagrams follow a set of fusion rules, which allow for obtaining many equivalent representations, simplifying the connections of complex spiders. Since 2019, much work has been done to optimize quantum circuits based on ZX diagrams. For instance, [49, 51, 96, 182] studied the optimization of the number of spiders in ZX diagrams; [44, 75, 94, 134] researched the optimization of T gate counts; and papers like [59, 92, 175, 193] introduced heuristic algorithms for the contraction and optimization of ZX diagrams. By optimizing the ZX diagram representation of quantum circuits, we can obtain more streamlined quantum circuits. When these ZX diagrams are mapped back to gate circuits, we can get circuits that meet the requirements. Compared to directly optimizing quantum gate circuits, the optimization methods based on ZX diagrams can introduce different characteristics, making them a relatively common method for representing quantum circuits.

3   Quantum Logic Circuit Synthesis

In this section, we address the challenge of quantum logic circuit synthesis. Running a quantum algorithm on a quantum processor necessitates translating the algorithm's unitary transformations into quantum circuits. Typically, these circuits are designed manually [87] or through decomposition algorithms [43, 97, 184]. This process demands substantial human labor and expertise, often resulting in circuits that are particularly deep. In the current Noisy Intermediate-Scale Quantum (NISQ) era, various constraints must be considered, including the accuracy of single and double qubit gates, qubit decoherence time, and the topological structure of physical qubits. Constructing a quantum circuit that achieves the highest possible accuracy while adhering to these constraints is a pressing challenge. Therefore, this section delves into methods for automatically generating quantum circuits that meet these requirements. This automated synthesis of quantum logic circuits is also referred to as the Quantum Architecture Search (QAS) problem in literature [48, 115, 209].

3.1 Problem Definition

First, we provide the mathematical definition of the QAS problem. The QAS problem originates from the Neural Architecture Search (NAS) problem, which automates the synthesis and optimization of neural network structures and parameters to replace human expert design. NAS has been widely applied in the field of computer vision, such as image classification [111] and object detection [188], and has achieved success on multiple datasets. Similar to the NAS setting, when defining the QAS problem, we first determine the set of candidate gates and the search space. Assuming we need to search for a circuit with n qubits and m layers, define the set of candidate gates as $\mathcal{G}$, and the size of the candidate set as $k = |\mathcal{G}|$, then the search space of the QAS problem is of the size $k^{n \times m}$. The candidate gate set $\mathcal{G}$ can include single-qubit, two-qubit, and parameterized quantum gates. We denote the selected candidate gates as matrix $\mathbf{M} \in \mathcal{G}^{n \times m}$, define $\mathbf{U}ij = \sigma(\mathbf{M}ij)$, where σ maps a quantum gate on the i-th qubit and j-th layer to a $2^n \times 2^n$ unitary matrix with all the irrelevant qubits as identity. The unitary matrix corresponding to the quantum circuit obtained through the search can be expressed as:

$$\hat{\mathbf{U}} = \prod_{j=1}^{m} \prod_{i=1}^{n} \hat{\mathbf{U}}_{ij} = \prod_{j=1}^{m} \prod_{i=1}^{n} \sigma(\mathbf{M}_{ij}). \tag{8}$$

For different applications, the QAS problem can be further divided into two versions. The first one searches the circuit with a target unitary matrix $\mathbf{U}$ and the corresponding optimization objective can be expressed as:

$$\min ||\mathbf{U} - \hat{\mathbf{U}}||_F, \tag{9}$$

where $|| \cdot ||_F$ represents the F-norm of the matrix. The second one designs the circuit based on a series of quantum initial states $|\chi_i\rangle$ and corresponding final states $|\psi_i\rangle$, without an explicit unitary matrix. The corresponding optimization objective can be expressed as:

$$\max \sum_i |\langle \psi_i| \hat{\mathbf{U}} |\chi_i\rangle|^2. \tag{10}$$

Both versions of QAS problems have extensive and important applications, which we will introduce next.

3.2 Applications of Quantum Architecture Search

3.2.1 *Quantum Architecture Search with Unitary Matrix*

The most important application for QAS with unitary matrix is the circuit implementation of oracles. Here, an oracle refers to a black-box unitary matrix often seen in quantum algorithms. We may not know its specific implementation yet, but we understand the unitary operation corresponding to this oracle. Quantum oracle is of fundamental significance in quantum computing and quantum information, but implementing an arbitrary quantum oracle requires exponentially many gates [141]. Taking the oracle in Grover's algorithm [64] as an example, we denote the oracle as $\mathcal{O}$, which can be represented as a unitary operation:

$$|x\rangle |q\rangle \xrightarrow{\mathcal{O}} |x\rangle |q \oplus f(x)\rangle, \tag{11}$$

where $|x\rangle$ is the register recording the index during the search, $\oplus$ is modulo 2 addition, and $|q\rangle$ is ancilla qubit. When $f(x) = 1$, the ancilla qubit will flip; otherwise, no operation is performed. Therefore, in the searching process of Grover's algorithm, we can prepare $|x\rangle |0\rangle$, then apply the oracle $\mathcal{O}$ and determine whether x is the solution to the search problem by checking if the ancilla qubit is $|1\rangle$.

Researchers typically manually design these oracles using basic quantum gates (i.e., quantum gates without parameters) [23, 158]. However, with the proposal of QAS methods, parameterized quantum circuits have also been used to search for implementations of these oracles more efficiently. Search algorithms are used to evolve quantum oracles [47] as well as design quantum adders [45, 108], which is further applied to quantum autoencoders [103]. Recent work [209] has also used QAS algorithms to design circuits for Quantum Fourier Transformations.

3.2.2   *Quantum Architecture Search without Unitary Matrix*

In practical applications, situations without a target unitary matrix are more common, especially in quantum machine learning problems such as Quantum Neural Networks (QNN) or Variational Quantum Eigensolvers (VQE). Theoretically, if there are enough linearly independent input-output pairs (quantum initial states and corresponding final states), the unitary matrix corresponding to the transformation circuit is mathematically solvable, making this situation equivalent to the case with a target unitary matrix. However, the search process becomes more challenging when there is a lack of linearly independent input-output pairs.

The VQE algorithm [152] is a notable application without a target unitary matrix, with extensive utility in quantum chemistry, combinatorial optimization, and quantum simulation. Its core functionality involves optimizing parameterized quantum circuits through classical optimizers to determine the minimum eigenvalue and corresponding eigenvector of a given Hamiltonian. VQE has garnered significant attention in the NISQ era due to its relatively shallow parameterized quantum circuits and adaptability to noisy environments on quantum processors. When utilizing VQE to solve quantum chemistry problems as the ground state energy estimation problem, the process typically involves discretizing electron distributions into orbitals through second quantization [21], mapping these orbitals to qubits using algorithms such as the Jordan-Wigner transformation [89], and employing unitary coupled cluster theory [179] to generate single and double excitation ansatzes for quantum state evolution. This procedure necessitates the derivation of distinct quantum circuits for each molecule or Hamiltonian, requiring both quantum chemistry expertise and specialized algorithms. Moreover, the resulting ansatzes are often deeper [17] due to the preservation of specific quantum chemical symmetries, rendering them challenging to implement on current NISQ hardware. Consequently, researchers have explored QAS methods to automatically design VQE ansatz, aiming to reduce design complexity and generate circuits that satisfy given constraints (e.g., circuit depth, two-qubit gate count, or physical qubit connectivity). The search process utilizes the same loss function as in the ground state energy estimation problem. Existing research has applied QAS methods to enhance unitary coupled cluster ansatzes [63, 165] or directly search for VQE circuits without prior ansatz knowledge [147, 187, 195]. Similarly, in combinatorial optimization problems, such as the maximum cut problem solved using the Quantum Approximate Optimization Algorithm (QAOA) [52], studies have proposed methods to optimize the original QAOA circuit [118, 119] or design related circuits that circumvent Ising model decomposition [50, 209].

QNN is another representative application without a target unitary matrix. Unlike VQE, which evolves a quantum state according to a given Hamiltonian to obtain the eigenvector corresponding to the minimum eigenvalue, QNN algorithms employ supervised learning. The inspiration comes from classical neural network design methods [65, 105], utilizing layered quantum gate structures to emulate classical neural networks. QNNs are trained to fit training set data, with the loss function typically defined as the prediction accuracy on the test set. For instance, quantum convolutional neural networks [37] and quantum recurrent neural networks [18] incorporate artificially designed quantum layers, which are implemented through parameterized gates and entanglement gates. Similar to VQE, QNNs employ parameterized quantum circuits, calculating the loss function and updating parameters via classical computers. Consequently, these parameterized quantum circuits can be directly obtained by QAS algorithms through the original loss function. QNNs are hypothesized to potentially offer faster training speeds and higher prediction accuracy compared to classical neural networks [83]. Recent studies [50, 187, 208] have utilized QAS algorithms to design complete circuits for QNNs or components of their layers, aiming to achieve improved training effects and modify certain fixed ansatz templates such as the Hardware Efficient Ansatz (HEA).

Beyond these applications, there are other potential applications without target unitary matrices, such as the design of quantum error correction codes and quantum nonlinear units. Quantum error correction codes are crucial in realizing fault-tolerant quantum computing by encoding quantum information to resist environmental noise [29]. The essence of quantum error correction codes involves encoding the original quantum state into a larger state through specific encoding methods to enhance information protection. Consequently, automated circuit synthesis algorithms can be introduced in the encoding and decoding processes of quantum error correction codes. Current research in this domain encompasses heuristic methods [160], machine learning approaches [33, 37], and reinforcement learning techniques [140, 205]. Similarly, the design of quantum nonlinear units is also a potential application area. Since quantum circuits contain only linear unitary transformations, it is difficult for quantum machine learning to introduce

Table 1   Review of quantum architecture search (quantum logic circuit synthesis) methods. "End-to-end" means the updating of both circuit structure and parameters are done by gradient descent. "Super-circuit" stands for pre-fixed circuit structure using human knowledge.

Paper	Method	Application	Sourced training	Structure Updating Method	Rotation Parameter Updating	Super-circuit Free	End-to-End	Circuit Noise
[192]	Genetic Algorithm	Quantum Teleportation	✗	Genetic Algorithm	✗	✓	—	No
[91]	Simulated Annealing	QFT	✗	Simulated Annealing	✓	✗	✗	IBM-Q, Rigetti
[202]	RL	Bell state	✗	Reward Function	✗	✓	—	Simulated
[101]	RL	Bell and GHZ state	✗	Reward Function	✗	✓	—	Simulated
[147]	RL	State Preparation	✗	Reward Function	✓	✓	✗	No
[150]	RL	State Preparation	✗	Reward Function	✓	✓	✗	IBM-Q
[208]	Machine Learning	Max-Cut, Classification	✓	Neural Network	✓	—	—	No
[71]	Machine Learning	State Preparation	✗	Sampling	✓	✗	✗	No
[209]	Machine Learning	Max-Cut, QFT, Error Mitigation	✗	Monte-Carlo Sampling	✓	✗	✓	Simulated
[48]	Machine Learning	State Preparation, Classification	✗	Sampling	✓	✗	✗	Simulated
[187]	Machine Learning	State Preparation	✗	Sampling	✓	✗	✗	IBM-Q
[195]	Machine Learning	State Preparation, Max-Cut, Classification	✗	Gumbel-Softmax	✓	✓	✓	Simulated

nonlinearity as conveniently as in classical machine learning methods [167]. Some recent studies have attempted to use linear oracles to approximate the results of nonlinear operations [159], thus achieving the effect of nonlinear layers to some extent. This makes the design of nonlinear units another potential application area for QAS algorithms.

3.3   Quantum Architecture Search Methods

We summarize the various QAS algorithms in Table 1. The table organizes the characteristics of QAS algorithms, particularly focusing on the methods for updating structural parameters and whether the updates of structural parameters and rotation parameters are based on gradient descent. Additionally, we indicate whether super-circuits are required during the search process and whether noise experiments are included. The specific algorithm classifications are as follows:

3.3.1   *Heuristic Algorithms*

Heuristic algorithms, particularly Genetic Algorithms (GAs), were among the earliest methods applied to automate quantum logic circuit synthesis [192]. Initial research focused on using GAs to search for simple circuits, such as quantum teleportation. Subsequent studies demonstrated the potential of GAs in evolving relatively simple quantum circuits, including quantum error correction codes and quantum adders [15, 103, 104, 125, 154]. The basic process of GAs involves: 1) selecting the initial population; 2) evaluating individuals in the current population; 3) sampling based on individual quality to form a candidate pool; and 4) performing crossover and mutation in the candidate pool. Steps 2 to 4 are repeated until a stop condition is met.

GAs are particularly suitable for problems where the solution is a sequence, as sequences can be viewed as strands of genes. From the quantum circuit representation methods introduced in the previous section, we can see that quantum circuits can be represented as sequences, with each gene locus corresponding to a quantum gate. Information such as gate type, parameters, and qubit index can be encoded as integer strings. The evaluation method in step 2 can refer to equations 9 and 10, while crossover and mutation correspond to modifying the position, type, parameters, and qubit information of quantum gates. The performance of GAs is influenced by factors such as initial population size, candidate pool size (number of sampled quantum circuits), and gene length (quantum circuit depth). Increasing the types of gates in the circuit further extends the encoding length for each quantum gate, adding complexity to the optimization process.

In addition to genetic algorithms, simulated annealing algorithms have also been applied to QAS [91]. This approach introduces parameterized quantum gates and randomly modifies parts of the existing optimal circuit structure during the search process. The internal parameters are then trained to obtain the optimal solution for the current structure. If this solution surpasses the existing optimal structure, an update is granted; otherwise, an update is made only with a probability that decreases exponentially with the performance gap. While simulated annealing algorithms show some effectiveness, they suffer from weak scalability. Expanding the scale requires pre-defining a circuit structure, with each modification necessitating training to optimality, resulting in significant computational overhead. Consequently,

heuristic algorithms have not found widespread application in the automatic synthesis and evolution of variational quantum circuits.

3.3.2 *Reinforcement Learning Algorithms*

To enhance the performance of QAS, researchers have explored machine learning algorithms beyond heuristic approaches, leveraging their powerful capabilities. Reinforcement learning (RL), as a representative example of unsupervised learning, has emerged as a suitable alternative, given the difficulty of introducing labeled input-output pairs in QAS. RL algorithms typically operate in discrete time steps, where an agent makes decisions at each step, and the environment provides corresponding rewards. The agent's goal is to maximize rewards through a limited set of actions. In QAS, the agent's action pool comprises candidate quantum gates, with each decision adding a gate to the circuit. The reward is calculated using a loss function after each addition. A key advantage of RL is its ability to generate circuits without preset maximum depths, theoretically allowing for continuous circuit deepening. Unlike heuristic methods, RL accommodates parameterized quantum gates, optimizing internal parameters while adding gates.

Primitive studies [101, 202] tested RL algorithms on two-qubit Bell states and three-qubit GHZ states. RL algorithms are then applied to automate the synthesis of variational quantum circuits for solving ground state energies [147]. Their experiments with four-qubit and six-qubit lithium hydride Hamiltonians assigned a -1 reward for each added gate and a large positive reward for achieving chemical accuracy (i.e., 1.6×10^{-3} Ha). This reward mechanism successfully guided the agent to find circuits achieving chemical accuracy, validating RL's feasibility. Recently, Patel et al. [150] introduced curriculum learning, enhancing the guidance mechanism towards chemical accuracy and demonstrating significant performance improvements over previous work.

3.3.3 *Sampling-based Learning Algorithms*

In addition to reinforcement learning, sampling-based learning algorithms represent another significant class of circuit synthesis methods. These algorithms initially define the search space, considering a quantum circuit with n qubits, m layers, and k candidate gates, resulting in a search space of size $k^{m \times n}$. The methodology proceeds as follows: first, sampling is performed on the $n \times m$ grid points to generate a complete quantum circuit. Subsequently, the sampled circuit undergoes evaluation, informing modifications to the sampling strategy for future iterations. This process is repeated iteratively to complete circuit synthesis. Notably, sampling-based learning algorithms demonstrate compatibility with parameterized quantum gates, allowing for optimization of internal circuit parameters post-sampling. The primary challenge inherent in this approach stems from the discrete nature of the sampling process, which precludes the production of differentiable gradients. Consequently, developing effective mechanisms for updating the sampling strategy remains a critical area of research in this domain.

Zhang et al. [208] proposed a neural network predictor for circuit synthesis, training it on small-scale quantum circuits and applying it to slightly larger-scale problems. This approach, while one of the few supervised learning models for QAS, faces limitations due to the diversity of quantum gates and the non-trivial relationship between gate configurations and circuit purposes. Recently, He et al. [71] introduced a training-free scheme based on [208], which reverted to sampling-based optimization. However, this study's evaluation scale was limited (up to six qubits) and lacked comparison with state-of-the-art algorithms.

Zhang et al. [209] employed Monte Carlo gradient estimation methods to update sampling strategies, demonstrating efficacy in Quantum Fourier Transform and error mitigation problems. Du et al. [48] selected the optimal circuit through multiple sampling and ranking, testing their algorithms on four-qubit hydrogen ground state energy prediction and image classification QNN design problems. Wang et al. [187] proposed a method similar to [48], focusing on noise adaptation for quantum hardware. This approach generates shallow quantum circuits under the constraints of the quantum hardware to mitigate errors. Experimental results on IBM quantum devices demonstrated significant improvements over manually designed deeper circuits. However, both [48] and [187] utilize pre-defined circuit modules (termed super-circuits or supernets), substantially reducing the search space through pre-designed modules or specified layer arrangements. This approach potentially limits the demonstration of the algorithms' true capabilities, as successful circuit design relies heavily on pre-provided key circuit modules.

Lu et al. [115] addressed this issue by questioning the efficacy of circuit structure sampling and synthesis and aligning algorithm applications. The study proposed using the arbitrary unitary approximation

problem to benchmark QAS algorithms, considering this task the most challenging in QAS. To accommodate potentially deep decomposition depths, a simplified version - arbitrary circuit reconstruction - was introduced. This task provides a unitary matrix with a known circuit solution within a certain depth, enabling fairer algorithm comparisons. Results indicated that after removing prior information, such as circuit modules, the performance of [48] and [209] decreased significantly. Notably, when candidate gates were limited to non-parameterized quantum gates, machine learning algorithms generally underperformed compared to heuristic algorithms.

To enhance the efficiency and accuracy of searching algorithms, Wu et al. [195] proposed a method utilizing Gumbel-Softmax [20,66,86] to bridge the gap between discrete sampling and continuous gradient updates. This method uses the argmax term in the forward pass and the softmax term in backpropagation, addressing issues inherent in sampling-based learning algorithms and significantly improving search efficiency. The study eliminates prior knowledge interference by searching directly within the complete $k^{m \times n}$ space. It evaluates the method on three problems: VQE circuits (ground state energy prediction and Max-Cut) and QNN circuits (image classification), demonstrating notable improvements. Furthermore, the article examines the scalability of QAS algorithms, proposing two search processes: Macro Search and Micro Search. Macro Search conducts global domain searches, while Micro Search uses preset sub-circuit modules with shared structural parameters but distinct internal rotation parameters. Unlike [48] and [187], which predetermine circuit module structures, Wu et al. [195] searches for the module structures. They facilitate the identification of frequently reused modules, such as single and double excitation operators in ground state energy estimation or quantum convolution and pooling modules in image classification. This approach reduces the search space and enables the application of modules discovered in small-scale problems (e.g., six-qubit lithium hydride) to larger-scale problems (e.g., 18-qubit methane). Given that [195] and [209] render the entire search process differentiable end-to-end, this approach can be categorized as a new track: differentiable quantum architecture search.

4   Quantum Logic Circuit Optimization

In the context of quantum computing, several unavoidable factors must be considered when executing quantum circuits, including qubit coherence time, environmental noise, etc. These constraints necessitate the optimization of quantum logic circuits to reduce their complexity. While quantum logic circuit synthesis creates circuits from scratch, optimization algorithms refine existing circuits based on specific targets. Consequently, circuits produced by synthesis algorithms can be further enhanced through optimization techniques. Moreover, by integrating the constraints of circuit synthesis with the optimization targets, we can develop a unified synthesis optimization algorithm. This approach enables the direct creation of optimal quantum logic circuits that meet current optimization objectives.

4.1   Problem Definition

We begin by presenting a formulation of the quantum logic circuit optimization problem. Analogous to the quantum logic circuit synthesis problem, we employ a unitary matrix to represent a given quantum circuit. Let $\mathbf{U}$ denote the unitary matrix of the original circuit to be optimized, and $\hat{\mathbf{U}}$ represent the unitary matrix of the optimized circuit. Given a set of optimization targets κ_i, where we assume that all targets are to be minimized, the corresponding optimization objectives can be expressed as:

$$\min \sum_i \kappa_i(\hat{\mathbf{U}})$$
$$\text{subject to } ||\mathbf{U} - \hat{\mathbf{U}}||_F \leqslant \varepsilon. \tag{12}$$

In this formulation, ε denotes the permissible unitary matrix error. When ε equals 0, the two circuits are required to be completely equivalent. The term $\kappa_i(\hat{\mathbf{U}})$ represents the score of the optimized circuit with respect to a specific optimization target. In the following sections, we will summarize and categorize the primary optimization objectives of quantum logic circuits, followed by a systematic review of various optimization methodologies.

Table 2   Quantum logical circuit optimization methods versus optimization target.

Methods \ Target		Gate Count	Circuit Depth	CNOT Count	T Count / T Depth
Brute-Force Search		$[99, 199]$	$[8, 99, 153]$	$[26]$	$[8]$
Pattern Matching	Peephole Optimization	$[27, 69, 121]$		$[112, 173, 198]$	$[137]$
	Recursion	$[14, 129]$			
	Divide and Conquer	$[156]$		$[149, 196]$	
	Commute	$[27, 76]$		$[136]$	$[2, 76, 207]$
	Template	$[27, 85, 122\text{–}124]$			
	Graph Contraction			$[176]$	$[9, 95, 176]$
Reduction		$[166]$		$[166]$	$[7, 9, 75]$
Machine Learning	Heuristic Algorithm			$[6, 42]$	
	RL	$[56, 110, 161]$	$[56, 110]$	$[110, 161]$	$[163]$

4.2   Quantum Logic Circuit Optimization Targets

The objectives of quantum logic circuit optimization algorithms are directly correlated to their optimization methods. Table 2 illustrates the optimization methods in various research and their corresponding optimization objectives. It can be seen that, excluding brute-force search and reinforcement learning, other methods are aimed at more specific optimization targets. Overall, there is an emphasis on optimizing gate count and CNOT gates, while optimization of T gates is closely related to phase, resulting in a comparatively limited array of available optimization methods.

4.2.1   *Gate Count*

Optimization of the number of gates represents a fundamental objective in quantum circuit design. This optimization remains crucial across the current NISQ era and prospective fault-tolerant quantum computing paradigms. In the NISQ context, where qubit coherence time is limited and noise effects are significant, minimizing the number of quantum gates serves to reduce overall circuit noise, thereby enhancing operational accuracy. Moreover, in the realm of fault-tolerant quantum computing, gate count reduction contributes to improved computational efficiency of quantum hardware. Thus, the imperative to minimize quantum gates persists as a key strategy for advancing quantum circuit performance, irrespective of the underlying quantum computing framework.

Various works such as $[14, 27, 56, 76, 99, 110, 122\text{–}124, 126, 129, 156, 166, 199]$ focus on optimizing quantum gate count in different types of circuits. Methods proposed by $[99, 156, 166]$ guarantee optimal Clifford circuits, with $[166]$ demonstrating scalability. We also have works targeting other gate sets, such as the parameterized gate set $[199]$, Toffoli networks $[122]$, Multi-Control Toffoli (MCT) circuits $[14, 129]$, and NCV circuits $[123, 124]$. There are also works focusing on reducing the gate count for VQE circuits $[126]$.

4.2.2   *Circuit Depth*

Circuit depth is intrinsically associated with the total runtime of a quantum circuit, which is constrained by the coherence time of qubits. Consequently, optimizing circuit depth holds practical significance for scalable quantum computing in the NISQ era. While reducing gate count and minimizing circuit depth are often considered alike, many studies address both objectives concurrently $[56, 99, 110]$. However, these goals are not entirely identical. Depth optimization aims to enhance circuit parallelism, thereby reducing overall execution time and improving accuracy and efficiency. A circuit with an optimal gate count does not necessarily imply optimal depth. For instance, a quantum circuit could be optimized to contain either 7 gates executed serially or 10 gates executed in 5 parallel layers. The former achieves the gate count optimization goal (7 gates), while the latter satisfies the depth optimization objective (5 layers). Therefore, this study distinguishes between these two optimization targets.

Circuit depth optimization is typically achieved by increasing the parallelism of quantum gates. Theoretically, a set of quantum gates without dependencies on each other can be executed in parallel. Dependencies in this context commonly refer to overlapping qubit operations. Thus, depth optimization can be understood as partitioning all gates in a circuit into the minimum number of mutually independent

gate groups. Numerous studies, including [8, 32, 56, 73, 74, 98, 99, 110, 153] have focused on circuit depth optimization. Specifically, Kliuchnikov et al. [99] achieved optimal circuits through recursion, a method that can be used to construct optimal circuit datasets. The algorithm presented in [8] extends beyond depth optimization with the capability of various goals. Other works focused on reducing the circuit depth of variational quantum circuits, especially QAOA circuits [32, 73, 74] and VQE circuits [98].

4.2.3 *CNOT Count*

In addition to general gate optimization, targeting specific quantum gate types for reduction can significantly enhance circuit performance under certain conditions. In the NISQ era, the implementation of two-qubit entangling gates, such as CNOT gates, presents greater challenges compared to single-qubit gates. This is due to longer execution times (single-qubit gate: 60 ns, CNOT gate: 660 ns) and higher error rates (single-qubit gate $\approx 1 \times 10^{-4}$, CNOT gate $\approx 1 \times 10^{-2}$)[2]. Consequently, minimizing the number of CNOT gates in circuit designs has emerged as a primary optimization objective, given that CNOT gates serve as representative two-qubit operations. This focused approach to gate reduction addresses the specific constraints and error profiles characteristic of current quantum hardware implementations.

Numerous studies [16, 39, 100, 132, 133, 169, 184] have analyzed and optimized the theoretical number of CNOT gates in quantum circuits by refining decomposition methods. These efforts have progressively reduced the theoretical upper bounds from $\mathcal{O}(n^3 4^n)$, as proposed in [16], to $\frac{23}{48} 4^n - \frac{3}{2} 2^n + \frac{4}{3}$ in [133]. Regarding CNOT gate depth, Jiang et al. [88] demonstrates that any n-qubit CNOT circuit can be optimized to $\mathcal{O}(\max\{\log n, \frac{n^2}{(n+m)\log(n+m)}\})$ depth using m auxiliary qubits. A comprehensive discussion and summary of recent advancements related to the theoretical number of CNOT gates is presented in [81].

Recent studies [6, 26, 42, 58, 110, 112, 139, 149, 161, 166, 176, 196] have focused on optimizing CNOT count in specific quantum circuits. Bravy et al. [26] identifies optimal Clifford circuits with minimal CNOT gates for up to 6 qubits, while Staudacher et al. [176] introduces methods to reduce CNOT count without compromising T gate optimization. Furthermore, optimizations tailored to variational quantum algorithms aim to enhance their performance in the NISQ era. Investigations such as [10, 35, 63, 68, 117, 131, 177, 181, 201, 210] focus on optimizing VQE circuits, whereas [118, 119, 215] concentrate on QAOA circuit optimization.

4.2.4 *T Count and T Depth*

Unlike optimization goals in the NISQ era, fault-tolerant quantum computing (FTQC) implementation costs are not solely limited to hardware considerations, but also depend significantly on the temporal and spatial overheads of error-correcting codes. In FTQC models, Clifford group elements often have relatively simple transversal implementations across many error-correcting codes [214], whereas non-Clifford elements (such as T gates) are more complex to implement. For instance, in Steane codes [4] and surface codes [57], T gates require magic state distillation to implement [25]. The space-time overhead for surface codes is analyzed in [30], where Clifford gates incur a space-time cost of $\mathcal{O}(c_T d^3)$ with a constant factor c_T of unit order, while T gates require $\mathcal{O}(C_T d^3)$ with $C_T \approx 160 \sim 310$. Here, spatial overhead refers to the number of physical qubits needed to encode a logical qubit, and temporal overhead denotes the duration of the encoding protocol [30]. The implementation cost of T gates significantly exceeds that of Clifford gates by orders of magnitude [30, 163]. Consequently, optimizing the quantity and depth of non-Clifford gates (T gates) is of paramount importance in FTQC.

Similar to reducing the overall gate count, directly minimizing the number of T gates is an intuitive approach. Several studies, including [2, 9, 75, 95, 137, 163, 176, 207], focus on optimizing T count. Abdessaied et al. [2] attempt to consolidate and reduce T count by reducing the number of H gates. Nam et al. [137] optimize T count for approximate Quantum Fourier Transform circuits. Kissinger et al. [95] and Heyfron et al. [75] employ the ZX diagram to reduce T count by extracting and consolidating non-Clifford phases. Amy et al. [9] transform the T count optimization problem into a minimum distance decoding problem for Reed-Muller codes, guaranteeing output circuits with optimal T count.

Parallel to efforts in optimizing circuit depth, several studies [7, 8, 207] have focused on exploiting T gate parallelism to minimize T gate depth in quantum circuits. These investigations primarily examine the effect of parallelizable T gates on reducing overall circuit operation time. Notably, Amy et al. [7]

---

2) Data source: IBM-Q superconducting quantum computer Kyoto

proposes a polynomial-time optimization algorithm that, in most instances, guarantees the resultant circuit exhibits minimal T gate depth.

4.3   Quantum Logic Circuit Optimization Methods

Based on the characteristics of different optimization methods, we summarize them in Table 3. The table lists the optimization targets and circuit representation methods for each optimization algorithm. It can be seen that gate circuits dominate, while phase polynomial representations are mainly used for T gate optimization. We also list some specific features, such as whether the algorithm requires ancillary qubits, whether it provides optimal solutions, and whether it optimizes parameterized circuits.

4.3.1   *Brute Force Search*

Brute force search algorithms find optimal circuits with specific depth and width through exhaustive enumeration. While this method guarantees optimal results, it incurs significant computational costs, limiting its application to small-scale logic circuits. Nevertheless, the optimal circuits generated by brute force search can be further utilized in other optimization algorithms. This section introduces the fundamental principles of brute force search algorithms.

The brute force search method typically comprises two core steps. First, a dataset containing all quantum logic circuits meeting specific criteria (e.g., certain depth and width) is constructed based on a set of candidate quantum gates. Subsequently, this dataset is traversed to identify the optimal circuit. As every subcircuit of an optimal quantum circuit is also optimal, the second step can employ bidirectional search or recursive backtracking to gradually identify the optimal quantum circuit composing the target unitary matrix from the dataset. However, these algorithms offer limited improvement in search efficiency. Therefore, investigating the optimization of dataset construction to reduce dataset volume and consequently decrease computational costs associated with searching is equally crucial.

Amy et al. [8] utilize a breadth-first search algorithm to construct a gate circuit tree, where each edge corresponds to adding a gate from the candidate set. A child node's circuit is derived by adding the gate on the edge to its parent node's circuit. When searching for the optimal circuit of depth k, all nodes at depth k in the tree are traversed. The study employs a bidirectional search method to reduce search depth, enabling the use of smaller datasets to search for larger circuits (using a dataset with maximum depth k to construct all optimal quantum circuits with depth not exceeding $2k$). The correctness of bidirectional search stems from the fundamental fact that subcircuits of optimal circuits are also optimal. Another contemporary study, [99], proposes using unitary matrix equivalence classes to reduce dataset size. Unitary matrices that are equivalent under logical qubit reordering are grouped into the same class, significantly reducing dataset size (one equivalence class contains $n!$ unitary matrices). Bravyi et al. [26] investigate the construction of equivalence classes based on CNOT depth, considering nine different scenarios for adding CNOT gates (accounting for various placements of single-qubit H and P gates). Unitary matrices that are equivalent after removing single-qubit gates at both ends and reordering logical qubits are grouped into the same class, storing one representative element. For a target unitary matrix, reduction is first performed to find a representative element with the same CNOT depth, from which the optimal circuit is then derived. This methodology enables the search for all optimal circuits within six qubits.

The optimization methods employed in [199] and [153] are analogous to brute force search. They first construct an optimal circuit dataset of specified size, then reduce costs by replacing local segments of the target circuit with optimal ones. However, this approach only guarantees local optimality, not global optimality of the entire circuit. The algorithm proposed in [153] initially defines an n-qubit, m-layer grid, randomly populated with quantum gates from a set. Equivalence classes are then constructed based on unitary matrix equivalence within a certain precision. For the target circuit, all replaceable subcircuits are identified, their unitary matrices computed, and optimal replacements with minimal depth are sought from the dataset's equivalence classes. Xu et al. [199] extends [153] from discrete to continuous quantum gate sets. The algorithm first constructs an optimal subcircuit dataset using breadth-first search and identifies equivalence classes for parameterized circuits. The research presents a verification method utilizing the function $|\langle\psi_0|\,\mathbf{U}(\mathbf{p}_0)\,|\psi_1\rangle\,|$, where $\mathbf{U}(\mathbf{p}_0)$ represents a quantum circuit with parameters $\mathbf{p}_0$, $|\psi_0\rangle$ and $|\psi_1\rangle$ are random quantum states, and $|\cdot|$ denotes complex modulus. By randomly sampling $|\psi_0\rangle$, $|\psi_1\rangle$, and $\mathbf{p_0}$, the function value of $\mathbf{U}$ can be calculated. If two unitary matrices yield identical results for one

Table 3   Review of quantum logical circuit optimization algorithms and characters.

Reference	Target	Circuit Representation					Ancilla qubits	Is Optimal	Parameterized Circuit
		Gate Model	Phase Polynomial	DAG	ZX Diagram	Tensor Network			
[156]	Gate Count	✓		✓			✗	✗	✗
[199]	Gate Count			✓			✗	✗	✓
[85]	Gate Count			✓			✗	✗	✓
[121]	Gate Count	✓					✓	✗	✗
[69]	Gate Count	✓					✓	✗	✗
[129]	Gate Count	✓					✓	✗	✗
[14]	Gate Count	✓					✗	✗	✗
[27]	Gate Count	✓					✗	✗	✗
[122]	Gate Count	✓					✗	✗	✗
[124]	Gate Count	✓					✗	✗	✗
[123]	Gate Count	✓					✗	✗	✗
[56]	Gate Count, Circuit Depth	✓				✓	✗	✗	✗
[99]	Gate Count, Circuit Depth	✓					✗	✓	✗
[110]	Gate Count, Circuit Depth, CNOT Count			✓			✗	✗	✗
[161]	Gate Count, CNOT Count				✓		✗	✗	✗
[166]	Gate Count, CNOT Count	✓					✗	✓	✗
[76]	Gate Count, T Count/Depth	✓					✗	✗	✓
[153]	Circuit Depth	✓					✗	✗	✗
[8]	Circuit Depth, T Count/Depth	✓					✓	✓	✗
[136]	CNOT Count	✓	✓	✓			✗	✗	✓
[26]	CNOT Count	✓					✗	✓	✗
[42]	CNOT Count	✓					✗	✗	✓
[196]	CNOT Count	✓					✗	✗	✓
[149]	CNOT Count	✓					✗	✗	✓
[198]	CNOT Count	✓					✗	✗	✓
[6]	CNOT Count		✓				✓	✗	✗
[173]	CNOT Count			✓			✗	✗	✓
[112]	CNOT Count	✓					✓	✗	✗
[176]	CNOT Count, T Count/Depth				✓		✗	✗	✗
[2]	T Count/Depth		✓				✗	✗	✗
[207]	T Count/Depth			✓			✓	✗	✗
[137]	T Count/Depth	✓					✓	✗	✓
[95]	T Count/Depth		✓		✓		✗	✗	✓
[75]	T Count/Depth					✓	✓	✗	✗
[163]	T Count/Depth					✓	✓	✓	✗
[7]	T Count/Depth		✓				✗	✗	✗
[9]	T Count/Depth		✓				✓	✗	✗

or more sets of variables, they are recognized as equivalent parameterized circuits. Utilizing this method, optimal replacement of certain local subcircuits within parameterized circuits can be implemented.

4.3.2 *Pattern Matching*

Pattern matching is a widely employed technique in quantum circuit optimization, designed to identify and replace known patterns within a circuit to enhance its efficiency. Although this approach typically focuses on optimizing local sub-circuits and may not guarantee global circuit optimality, it offers advantages of low time complexity and favorable scalability. In quantum circuit optimization, pattern matching typically involves predefined equivalence substitution rules. The basic process of pattern matching algorithms is to set up a group of sub-circuit substitution rules in advance. These rules are then applied by traversing through sub-circuits using methods such as circuit partitioning or sliding windows. When applicable, these substitution rules are executed to reduce the cost of sub-circuits.

Peephole Optimization: Peephole optimization, a well-established technique in classical compilation optimization [127], focuses on small local segments such as a few instructions in classical computing or a limited number of quantum gates in quantum logic circuits. In the context of quantum circuit optimization, peephole optimization involves traversing the circuit to identify optimizable local patterns. We first introduce a special form of peephole optimization based on qubit states [112]. This method analyzes qubit states using a quantum state annotation (analogous to finite state machines in classical compilers). During circuit execution, optimization is applied when a qubit is found in a predefined pure or basis state and a local sub-circuit matches a predefined pattern. This algorithm is distinctive in its alteration of the optimized circuit's unitary matrix while preserving functionality. Maslov et al. [121] and He et al. [69] optimize multi-control NOT gates by substituting Toffoli gates with phase Toffoli gates to reduce implementation costs. These two phase Toffoli gates differ from the standard Toffoli gate by a phase factor when acting on certain quantum states. However, due to their even times of applications, these phase factors cancel out in the final circuit. Consequently, the global effect is equivalent to that of the Toffoli gate, but with a reduced implementation cost. Sivarajah et al. [173] incorporates peephole optimization as a subroutine within the t $|ket\rangle$ compiler, focusing on efficient optimization of specific quantum patterns. Bravyi et al. [27] introduces symbolic peephole optimization, addressing limitations in applying pattern matching optimizations where the replacement sub-circuit must be fully decoupled from other circuit components. This method decomposes CNOT gates into projection operators on control qubits and symbolic Pauli operators on target qubits. Symbolic Pauli operators include exponential term labels indicating control qubit indices, optimizing the decomposed circuits using dynamic programming, and predefined circuit equivalence rules before restoration.

Recursive Decomposition: In addition to iterative pattern searching for optimization, recursive methods can also be employed to identify patterns within circuits. Recursive optimization is primarily applied to multi-controlled NOT gates. The core concept involves decomposing a controlled-NOT gate circuit with c control qubits into circuits containing controlled-NOT gates with c_1 and c_2 control qubits, where c_1 and c_2 satisfy $c_1 + c_2 = c$. A common decomposition approach sets $c_1 = c - 1$ and $c_2 = 1$, effectively reducing one control qubit per iteration. In [14], a controlled-NOT gate with c control qubits is recursively replaced by a controlled-NOT gate with $c - 1$ control qubits and CNOT gates, utilizing quantum Karnaugh maps to verify circuit equivalence. Miller et al. [129] adopt a more generalized decomposition strategy, breaking down a controlled-NOT gate with c control qubits into controlled-NOT gates with c_1 and c_2 control qubits. It then iterates through all possible (c_1, c_2) value pairs, generating $c-2$ combinations and selecting the most cost-effective one. This recursive approach ultimately determines the optimal implementation.

Divide-and-Conquer: Divide-and-conquer optimization first divides the complete circuits into sub-circuits of specified depth and width. These sub-circuit modules are then optimally reconstructed and subsequently merged. The QGo algorithm proposed in [196] focuses on optimizing CNOT gates (including SWAP gates) in quantum circuits with logical-to-physical qubit mapping. To partition an n-qubit circuit into multiple independent k-qubit circuit blocks (where gates in each block operate only on the corresponding k qubits, with module depth not exceeding the complete circuit depth), QGo employs a greedy heuristic algorithm. It first constructs a DAG-based quantum gate dependency graph, where dependencies arise from the order of gate operations on qubits. Subsequently, it traverses all possible circuit partitions, scoring each using a CNOT count-based heuristic function. The highest-scoring partition is greedily selected, and its gates are removed from the current circuit. This process continues until the

circuit is empty, signifying completed partitioning. QGo then employs a known optimal circuit synthesis algorithm [41] to reconstruct the decomposed circuit modules. Finally, all optimized circuits are merged, with potential gate eliminations at block interfaces. Prasad et al. [156] employs a similar divide-and-conquer approach to [196], but utilizes a linear, array-like data structure to describe quantum circuits, which can be viewed as the result of a depth-first search traversal of the DAG. Building upon [196],Patel et al. [149] proposes the QEst method, which replaces precise synthesis with approximate synthesis when composing optimal sub-circuits, achieving simpler circuit implementations within acceptable error margins.

Commutation: Commutation-based methods involve rearranging commutable gates to group specific gate types together, followed by other optimization techniques. Both [2] and [136] focus on optimizing the Clifford+T set. When optimizing T gates based on phase polynomials, H gates interfere with phase merging. Thus, these investigations aim to optimize T gates by reducing the number of H gates. Both approaches seek gates that commute with H for swapping, as consecutive H gates can cancel each other out. Clustering H gates together can reduce their overall count in the circuit, facilitating phase reduction. Due to the high computational complexity of determining commutativity between a quantum gate and a subcircuit, Nam et al. [136] propose a set of sufficient (but not necessary) rules to assess gate-subcircuit commutativity. The optimization method in [27] leverages gate commutation rules to relocate Pauli and SWAP gates, partitioning the circuit into three sections: Pauli gates, SWAP gates, and computational components (including CNOT, H, and S gates), allowing for tailored optimization strategies for each section. Diverging from direct quantum logic circuit optimization, Hietala et al. [76] introduce SQIR (Small Quantum Intermediate Representation), a characterization method between gate circuits and quantum assembly language. SQIR enables more efficient quantum program verification and equivalence checking. The VOQC optimization program proposed in this study utilizes a set of commutable gate exchange rules, combined with known single-qubit and two-qubit gate elimination rules, to optimize circuits represented in SQIR.

Template Matching: Template matching is a specialized extension of pattern matching, wherein we consider a sequence of quantum gates that collectively result in the identity transformation (i.e., the unitary gate $\mathbf{I}$). In this context, the first half of this sequence can be represented by the inverse of the second half of the gates. The formal definition of template matching is as follows: suppose a template consists of m quantum gates, which can be formulated as $\prod_{i=1}^{m} \mathbf{U}_i = \mathbf{I}$. Assume that in the quantum circuit yet to optimized, there exists a segment that matches the first $k \geqslant \lfloor \frac{m}{2} \rfloor + 1$ gates of this template. Consequently, this segment can be replaced by:

$$\mathbf{U}_1 \mathbf{U}_2 \cdots \mathbf{U}_k \rightarrow \mathbf{U}_m^{-1} \mathbf{U}_{m-1}^{-1} \cdots \mathbf{U}_{k+1}^{-1}. \tag{13}$$

Given that $k \geqslant \lfloor \frac{m}{2} \rfloor + 1$, the resulting circuit after substitution is guaranteed to be shorter than the original circuit.

Maslov et al. [122] employed template matching to optimize circuits containing NOT, CNOT, Toffoli, and MCT gates. They explored all templates of length less than or equal to 7 and proposed strategies for optimization by traversing circuits bidirectionally to identify matching segments. Similar methodologies were applied in [124] with optimizations tailored to specific gate sets. They also categorized short templates satisfying additional conditions into specific circuit rules, such as gate commutativity. Additionally, Maslov et al. [123] introduced two template matching-based algorithms to optimize total gate count and circuit depth. The gate count optimization algorithm initiates with smaller templates, traverses the circuit from left to right, and identifies replaceable sub-circuits in either direction. The depth optimization leverages gate commutativity to enhance the parallelism of gates on different qubits, thereby reducing circuit depth. Bravyi et al. [27] also utilized gate commutativity and template matching for circuit optimization. They segregated CNOT, H, and S gates using commutativity before applying template matching to optimize this circuit segment. Iten et al. [85] employed canonical forms of circuits (representing them as DAGs) to identify small templates or maximal template matches in large circuits. They utilized bidirectional matching, considering whether nodes in the graph's predecessor or successor DAG could match specific templates. While greedy matching suffices for optimal solutions in forward matching, introducing nodes in backward matching may induce errors in forward matching. Consequently, they initially constructed a gate selection tree for backward matching before searching for optimal solutions within this structure.

Graph Contraction: Graph contraction methods are primarily used in the optimization of circuits represented by ZX diagrams. These methods target specific patterns in a local region of the ZX diagram.

Several representative patterns include: 1) *Local Complementation*: Merging elements with special angles (e.g., $\pm\frac{\pi}{2}$) into adjacent angles. 2) *Pivoting*: Combining multiples of 0 or π through rotation into adjacent nodes. 3) *Elimination and Fusion*: Elimination of Empty Units and Fusion of Adjacent Units in Clifford Circuits. Kissinger et al. [95] focus on optimizing phases in ZX diagrams to reduce T count after circuit decomposition. For non-Clifford phases in the circuit, phase gadgets are introduced to represent phases as separate units apart from the non-phase circuit component. These gadgets are isolated from the non-phase part by H gates, and subsequently, the ZX diagram undergoes rule-based contraction optimization to derive an equivalent ZX diagram with optimized phases. Staudacher et al. [176] propose the first method to minimize CNOT count without compromising T count and depth optimization. The research introduces a heuristic scoring system for local complementations, rotation, and reduction rules, enabling random or greedy selection of optimization strategies based on post-decision scores until circuit complexity cannot be further reduced. To optimize the CNOT count, the scoring system directly correlates with the number of connecting lines in the diagram, allowing for adaption to different optimization objectives by modifying the scoring criteria. The study emphasizes that indiscriminate fusion of adjacent units may not always yield positive outcomes; conversely, dismantling units might facilitate the execution of new rules.

4.3.3 *Reduction*

An alternative approach to addressing optimization problems with specific objectives involves reducing them to other problems and employing specialized solvers. A prevalent method utilizes phase polynomials to represent quantum circuits, subsequently converting these polynomials into Reed-Muller codes. This transformation effectively reduces the circuit optimization problem to a Reed-Muller decoding problem, enabling the application of dedicated solving techniques. In optimizing for T gates, phase polynomials are frequently utilized to characterize circuits. In a phase function, as shown in Equation 6, introducing a common factor of $\frac{\pi}{4}$ ensures that each coefficient $\theta_i \in \mathbb{Z}_8$, where the coefficients represent the rotation parameters of Rz gates in the range $[0, 2\pi]$. Even coefficients indicate that the current phase can be realized using S gates, meaning the number of T gates directly correlates with the number of odd coefficients. When these coefficients are further reduced modulo 2, where 1 and 0 represent odd and even coefficients, respectively, we seek a set $\delta\boldsymbol{\theta} \in \{0,1\}^{2^n-1}$ such that $\boldsymbol{\theta} \oplus \delta\boldsymbol{\theta}$ is equivalent to $\boldsymbol{\theta}$ in the circuit (where $\oplus$ denotes bitwise XOR), and the Hamming weight of the new circuit is minimized, i.e., the number of 1s is minimized. This problem is equivalent to finding a binary code in an equivalence class that has the smallest Hamming distance from $\boldsymbol{\theta}$. Amy et al. [9] demonstrate that for circuits containing only T and CNOT gates, the required code size is $\mathcal{RM}(n-4, n)^*$, providing comprehensive theoretical proofs for related complexity bounds. Heyfron et al. [75] propose a compilation optimization paradigm for Clifford+T sets, specifically targeting T gate optimization. Initially, non-Clifford phases are extracted by introducing redundant qubits, isolating the Clifford parts in the circuit. The optimization of non-Clifford phases is reduced to RM decoding and low-rank decomposition problems. Prior to reducing to RM code decoding, Amy et al. [7] also attempt to reduce the optimization problem for T gates to quasi-matrix partitioning problems, using the same problem setup as [9] but constructing the phase function set as a finite quasi-matrix, thereby reducing the problem of generating optimal T depth circuits to finding minimal quasi-matrix partitions.

Another commonly used method is to reduce the circuit optimization problem to a SAT problem for solving. Schneider et al. [166] utilized the conclusion from [1] that an optimal Clifford circuit requires at most $\mathcal{O}(\frac{n^2}{\log n})$ quantum gates, and mainly focused on the synthesis and optimization of Clifford circuits. Assuming the optimal circuit contains m gates, where $0 < m \leqslant c\frac{n^2}{\log n}$ and c is a constant, at each step k, $0 < k \leqslant m$, there are $1 + 2n + 2 \times \frac{n(n-1)}{2} = 1 + n + n^2$, i.e., $\mathcal{O}(n^2)$ ways to add quantum gates to an existing circuit. This implies a total of $m(1 + n + n^2)$, i.e., $\mathcal{O}(\frac{n^4}{\log n})$ binary variables are needed to represent the composition of an optimal Clifford circuit, thereby transforming the circuit optimization problem into a SAT problem for resolution. To ensure the optimal m, a binary search strategy can be employed to iterate through the possible values of m, given that m ranges from 1 to $\mathcal{O}(\frac{n^2}{\log n})$. Initially, set m to the midpoint of this range; if a SAT solver returns a result, set the upper bound to m; otherwise, set the lower bound to $m + 1$. This method effectively determines the optimal m. Moreover, the algorithm can also enhance SAT problem constraints for CNOT gate optimization.

4.3.4 *Machine Learning*

In complement to traditional algorithms, machine learning and associated techniques have been incorporated into quantum logic circuit optimization. Although machine learning algorithms may not provide theoretical guarantees on solution accuracy, they demonstrate capability in addressing larger-scale problems, offering feasible solutions at reduced computational costs. This attribute imparts practical value to quantum logic circuit optimization algorithms, expanding their applicability in more complex scenarios.

Heuristic Algorithms: The fundamental principle of heuristic algorithms is to search the solution space through a series of rules and heuristic strategies to find near-optimal solutions. Compared to traditional exact algorithms, heuristic algorithms are generally more efficient, especially when dealing with quantum logic circuit optimization problems with large solution spaces, offering near-optimal solutions at lower computational costs. Amy et al. [6] proposes a relatively primitive heuristic algorithm to optimize CNOT counts in phase polynomial representations. It constructs a $0, 1$ matrix based on XOR strings in the phase function (see equation 6), where each column represents an XOR string from the phase function, with matrix elements set to 1 for corresponding qubits in the XOR string. Adding CNOT gates causes XOR flips in the control qubit row based on the target qubit row. At each step, the algorithm greedily selects the qubit pair that causes the most flips to add a CNOT gate. If a column has only two 1, it deterministically adds that CNOT gate. This constructs a pure CNOT circuit and analyzes the optimal CNOT count solution. Davis et al. [42] employs the A^* algorithm to optimize CNOT counts in circuits. Given a set of quantum gates, allowed precision error, and maximum CNOT count, the algorithm starts from an empty circuit. Candidate circuits are stored in a priority queue. In each iteration, the algorithm pops the top circuit, attempts to append gates from the given set to the circuit's end, calculates the distance to the target unitary, and checks if it's within the allowed error range or if the CNOT count exceeds the limit. If requirements are met, it returns the current circuit and terminates. If the queue is empty, no solution exists. The comparison function of the priority queue is the same as the total cost function of A^* algorithm, comprising the cost of the existing path and a heuristic estimate of the additional cost needed to reach the target unitary.

Reinforcement Learning: In quantum logic circuit optimization, reinforcement learning offers a significant advantage over other algorithms by automatically discovering novel or undefined optimization strategies and matchable patterns. Fosel et al. [56] present a preliminary algorithm that encodes quantum gates as triplets within a three-dimensional tensor representation of quantum circuits, using qubit indices, gate types, and depth as coordinates. The algorithm incorporates predefined hard rules (transformations with direct benefits, such as adjacent gate cancellation) and soft rules (transformations with indirect potential benefits, like swapping adjacent commutable gates). During each iteration, the agent is limited to deciding on soft rule usage, while hard rules are applied to the maximum extent for optimal benefit. Agent actions are tensor-encoded, with single-gate rules encoded by depth and qubit index and two-gate rules by the minimum common qubit index and the depth of the first gate. The reinforcement learning-trained agent employs a convolutional neural network, taking the circuit's 3D tensor representation as the input state and outputting the policy (encoded circuit transformation actions) and the current state value (indicating optimization potential). Rewards are based on cost changes, such as the reduction in the CNOT count, before and after each transformation.

Li et al. [110] replace manually specified action sets with a policy network. It first converts gate circuits into graph structures and employs a graph neural network to extract quantum gate representations. The policy network includes quantum gate and transformation selection, incorporating probability distribution-based sampling with gradient updates via temperature-coefficient softmax. Experimental results show significant improvements over existing compilation optimizers. Riu et al. [161] introduce a reinforcement learning method for ZX calculus, using a policy network in combination with a graph neural network to identify replaceable positions in ZX diagrams. Actions involve substituting the output positions from the policy network, where the substitution concentrates on common equivalent circuit simplifications for the Clifford set. Rewards are based on gate count, necessitating the conversion of ZX diagrams to quantum circuits. Delayed conversion strategies are employed to reduce computational overhead. Ruiz et al. [163] optimize T count by transforming the problem into the low-rank decomposition of three-dimensional tensors, adopting Google's AlphaTensor method [53]. It employs phase polynomials up to the third order, corresponding to T gates, Controlled-S gates, and Controlled-Controlled-Z gates. By encoding these coefficients into a third-order tensor, the problem is reframed as the optimal decomposition of the circuit's third-order tensor. Additional pattern matching strategies for T and Toffoli gates are

employed to streamline consecutive circuit segments. The study highlights that reinforcement learning algorithms can spontaneously discover novel matching patterns, complementing the manually designed strategies.

4.3.5 *Circuit Optimization for Variational Quantum Algorithms*

Variational Quantum Algorithms (VQAs) are considered among the most promising candidates for demonstrating quantum supremacy in the NISQ era, owing to their shallow circuit depth and the adaptability to circuit noise and other factors. Consequently, numerous quantum circuit optimization algorithms have been developed, targeting specific VQAs and corresponding quantum hardware characteristics. Several comprehensive reviews have explored related issues [24, 31, 54, 180]. This section presents a concise overview of these recent developments in the field.

Variational Quantum Eigensolver (VQE): A key application of VQE is solving ground state energies in quantum chemistry problems. Common algorithms include Hardware-Efficient Ansatz (HEA) [90] and Unitary Coupled Cluster (UCC) [162]. HEA circuits adhere to hardware constraints and are simple but less accurate, while UCC methods are precise but complex due to creation-annihilation operator implementations. Continuous efforts have been focused on simplifying these operators in circuits. ADAPT-VQE [63] reduces operator usage by incrementally building ansatz circuits, selecting operators with the highest energy gradient from a pre-determined pool. This accelerates VQE convergence and significantly reduces the number of required operators and circuit depth. Qubit-ADAPT-VQE [177] implements a hardware-friendly ADAPT-VQE variant to minimize SWAP gates in existing quantum hardware topologies. QEB-ADAPT-VQE [204] uses Qubit-excitation evolution instead of Fermionic-excitation, allowing fixed structures for single and double excitation operators. It outperforms FEB-based algorithms [63,177] in CNOT count, circuit parameter count, and algorithm iterations. Magoulas et al. [117] combines FEB and QEB methods, employing an approximation strategy to balance CNOT gate reduction and accuracy preservation. DISCO-VQE [28] achieves more compact ansatz circuits by combining spin-symmetry-preserving fermionic operator products with a global optimization algorithm, simultaneously optimizing discrete operators and continuous variational parameters. TETRIS-ADAPT-VQE [10] accelerates ansatz synthesis by introducing multiple non-interfering operators per iteration, contrasting with ADAPT-VQE's single-operator approach. Diverging from ADAPT methods, Halder et al. [68] incorporates Restricted Boltzmann Machines (RBM) [5, 106, 164] and many-body perturbation theory in ansatz construction. It uses low-level excitation UCC operators as the RBM training set to generate higher-level excitation operators, achieving high accuracy while maintaining low complexity (e.g., UCCSDT accuracy with UCCSD-level circuit complexity).

Quantum Approximate Optimization Algorithm: The Quantum Approximate Optimization Algorithm (QAOA), introduced by Farhi et al. [52], plays a crucial role in solving quadratic unconstrained binary optimization (QUBO) problems. Alam et al. [3] proposes four general methods for optimizing QAOA circuits through gate commutation. Focusing specifically on QAOA circuits designed to solve the Max-Cut problem, Arufe et al. [11] presents a decomposition-based genetic algorithm for circuit optimization. Majumdar et al. [119] introduces two optimization strategies based on edge coloring [186, 190] and depth-first search, respectively, yielding optimizations of $\lfloor \frac{n}{2} \rfloor$ and $n-1$ CNOT gates (where n represents the number of vertices in the original graph). Majumdar et al. [118] further improves upon the DFS-based strategy from [119], proposing a heuristic greedy algorithm that reduces circuit depth while maintaining the $n-1$ CNOT gate optimization. Inspired by the ansatz construction in ADAPT-VQE [63, 177], ADAPT-QAOA [215] selects operators with the highest energy gradient from an operator pool in each iteration, accelerating algorithm convergence. ADAPT-QAOA adapts to various hardware constraints, increasing the approximation ratio while reducing circuit parameters and CNOT count. Furthermore, Dynamic-ADAPT-QAOA [200] builds upon ADAPT-QAOA by eliminating layers with optimal parameters approaching zero, further reducing QAOA circuit size. Herrman et al. [74] approach QAOA circuit optimization from the perspective of enhancing combinatorial optimization problems, providing a global optimization method.

5   Qubit Mapping and Routing

To execute quantum circuits on quantum computers, logical qubits in the circuit must be mapped to the physical qubits of the quantum processor. Since physical qubits are not fully connected, two-qubit (and

Figure 3 Quantum Circuit Compiling. (a) An example qubit topology of a superconducting quantum processor, we select four working qubits (marked as grey); (b) Logical circuit; (c) After mapping logical qubits q_0, q_1, q_2, q_3 to physical qubits Q_9, Q_6, Q_{10}, Q_{14}, three SWAP gates are inserted since only neighbor qubits are connected; (d) Circuit after qubit routing which reduce the number of SWAP gates to only one.

multi-qubit) gates in the circuit may not be directly deployable after mapping, necessitating the introduction of additional SWAP gates to satisfy connectivity constraints. SWAP gates exchange information and states between two qubits, enabling two-qubit gates between non-adjacent qubits. Implementing a SWAP gate requires three CNOT gates. In the NISQ era, CNOT gates still exhibit significant noise, and the excessive introduction of SWAP gates can severely affect execution results. Therefore, minimizing the number of SWAP gates introduced is crucial. Figure 3 illustrates an example where, based on the given superconducting quantum topology, physical qubits Q_9 and Q_{10} cannot be connected, preventing the direct application of two-qubit gates between them. When compiling the logical circuit shown in Figure 3(b), three SWAP gates must be added to exchange qubits, ensuring the circuit satisfies the connectivity in Figure 3(a) (the last SWAP is omitted as the measurement step is unaffected by SWAP gates). After optimizing qubit routing, the circuit can be simplified to use only one SWAP gate, as shown in Figure 3(d).

Problem Definition: The Qubit Mapping problem can be uniformly defined as: given a quantum circuit, the coupling graph of the quantum computer, and noise conditions, determine the mapping relationship between logical and physical qubits, such that the mapped circuit satisfies connectivity constraints while minimizing the mapping's impact on circuit execution results.

5.1   Qubit Mapping and Routing Targets

5.1.1   *Gate Count*

In the NISQ era, quantum gate noise is non-negligible. To minimize overall circuit noise, reducing the number of quantum gates in the mapped circuit is essential. It should be noted that various studies [38, 77, 107, 109, 113, 114, 130, 144, 145, 148, 172, 191, 212, 213, 216, 217] may differ slightly in their approach to optimizing quantum gate count. These approaches can be broadly categorized into optimizing the total quantum gate count, the number of added quantum gates, SWAP gate count, and CNOT gate count. Notably, quantum gates added in the compiling stage are mainly the SWAP gates, and SWAP gates require only CNOT gates to implement. Thus, the above-mentioned objectives are unified as optimizing gate count.

5.1.2   *Execution Time*

Circuit execution time represents time performance, reflecting circuit execution efficiency, and also affects circuit execution results. Due to the physical qubit decoherence problem, longer circuit runtime increases the likelihood of qubit decoherence, leading to errors in circuit execution results. Consequently, several studies [135, 144, 206] consider circuit execution time during circuit mapping, aiming to minimize mapped circuit execution time while satisfying connectivity constraints.

5.1.3   *Circuit Depth*

This optimization objective usually appears in conjunction with the number of quantum gates. It reflects both the number of quantum gates and, to some extent, circuit execution time. Generally, deeper circuits imply longer execution time and increased likelihood of decoherence. Consequently, circuit depth is often included in the optimization objectives of quantum circuit mapping [38, 107, 113, 114, 138, 155, 171, 217].

5.1.4 *Fidelity*

Fidelity measures the similarity between two quantum states and can be used to assess the accuracy of circuit execution results by comparing the output quantum state to the theoretically expected state. However, direct qubit state measurement is challenging, making fidelity calculation on quantum computers difficult. Therefore, most studies [113, 114, 135, 144, 145, 178] utilize the PST (Probability of a Successful Trial) index to reflect fidelity, representing the probability of obtaining correct execution results, which can be directly calculated from measurement outcomes. Alternatively, [13] employs Kraus operators to simulate quantum gate noise models and qubit decoherence time, enabling direct calculation and optimization of execution result fidelity.

5.1.5 *Turnaround Time*

Nowadays, companies like IBM have released their quantum resources to the public. People can use their quantum computers for free on their quantum cloud. However, with the growing need for quantum resources, one may queue up on the cloud for a long time to run their quantum circuits. To fully utilize the limited quantum resources, Das et al. [40] propose multi-programming, i.e. executing multiple quantum circuits on a quantum computer concurrently. Wu et al. [194] further formulate it as a scheduling problem. An important objective in this problem is to minimize the average turnaround time (i.e. time cost from submission to completion of a quantum job), given the coupling graph, noise conditions, and quantum jobs in the queue.

5.2 Quubit Mapping and Routing Methods

The quantum circuit mapping problem can be further divided into two sub-problems: initial mapping and routing. Initial mapping determines the one-to-one correspondence between logical and physical qubits at the circuit's outset, while routing ensures that two(multi)-qubit gates in the circuit satisfy connectivity constraints by inserting SWAP gates or employing other methods. Specific optimization approaches can be categorized as follows:

5.2.1 *Optimal Solution*

Siraichi et al. [172] present a method to obtain an exact solution based on dynamic programming. Assuming a circuit with n qubits and m two-qubit gates, the solution to the subproblem is defined as $S(l, i)$, representing the minimum number of SWAP gates required to satisfy the constraints of the first i two-qubit gates with the final qubit mapping being l. The solution to the original problem can thus be derived from various subproblem solutions. While this method yields the optimal solution, its time complexity of $\mathcal{O}(n!^2 \cdot n \cdot m)$ significantly limits its scalability.

5.2.2 *Heuristic Algorithms*

This approach, prevalent in existing work, generally plans SWAP gate paths using noise conditions and inter-qubit distances to find near-optimal solutions efficiently. As quantum circuits can be converted into DAGs, some studies apply classical graph algorithms to solve this problem. Zulehner et al. [217] calculate local optimal solutions between circuit layers using the A* algorithm, incorporating a look-ahead mechanism to consider subsequent two-qubit gate impacts, approximating the global optimal solution. Zhang et al. [206] also employ the A* algorithm but consider individual gate execution time to optimize overall circuit execution time. Tannu et al. [178] utilize the Dijkstra algorithm for an approximate solution. To further reduce search space and improve solution quality, Li et al. [107] introduce a scoring function guiding heuristic search, determined by the weighted sum of distances between all two-qubit gates in current and look-ahead layers. The heuristic search selects SWAP gates, minimizing this scoring function. Building on this, several studies [113, 144, 145, 148, 216] have proposed improvements. Zhu et al. [216] first select SWAP gates reducing current layer qubit distances as candidates, then choose those maximally reducing look-ahead layer qubit distances. Niu et al. [144] incorporate noise and execution time terms into the distance function and introduce BRIDGE gates as SWAP alternatives. In [145], they further consider SWAP and BRIDGE gate noise, enhancing scoring function realism. Liu et al. [113] incorporate circuit optimization, reducing CNOT gate count through two-qubit block optimization and multiple two-qubit gate cancellation. Park et al. [148] add distance-based weights to each two-qubit gate term in the scoring function, assigning lower weights to longer distances.

5.2.3 *Reduction*

The quantum circuit mapping problem can be reduced to classical problems and solved using the corresponding solvers. Murali et al. [135], Wille et al. [191], and Molavi et al. [130] transform this problem into a Boolean Satisfiability Problem, while respectively employing SMT (Satisfiability Modulo Theories) solver, SAT solver, and MaxSAT solver to obtain solutions. Additionally, this problem can be reduced to a binary integer programming problem and solved using the commercial solver CPLEX [138]. As quantum circuits can be converted into DAGs, the initial mapping problem can be viewed as a Subgraph Isomorphism Problem. Ash-Saki et al. [13] first determine the graph structure of the initial mapping, then select the least noisy subgraph from all isomorphic subgraphs on the quantum computer topology as the initial mapping. Li et al. [109] consider the subgraph formed by all two-qubit gates in the top layer and find an isomorphic subgraph as the initial mapping. Due to the difficulty of achieving accurate subgraph matching, Park et al. [148] only require finding an approximation of the target subgraph structure as the initial mapping, while minimizing the difference between the two.

5.2.4 *Machine Learning*

The quantum circuit mapping problem can be viewed as a decision problem, making it amenable to reinforcement learning approaches. Pozzi et al. [155] define the state as the current circuit layout and target mapping, actions as a series of parallel executable SWAP gates, and rewards determined by the number of plannable gates after action. They employ a simulated annealing algorithm for action search, with the action value function approximated by the Deep Q Learning algorithm. Sinha et al. [171] augment the state with unplanned gates and utilize the Monte Carlo tree search algorithm for action search, estimating the value function with a graph neural network. Reinforcement learning can also be applied exclusively to initial mapping. Huang et al. [77] employ a multi-head attention mechanism to encode the circuit and decode it to obtain the initial mapping, then recursively extract sub-circuit structures and use the A* algorithm for routing. They use the number of gates added after circuit mapping as the reward.

6 Summary and Outlook

This survey comprehensively summarizes methods related to quantum logic circuit design and compilation optimization, encompassing quantum circuit representation, synthesis, optimization, qubit mapping, and routing. To execute quantum programs on NISQ-era quantum processors, quantum algorithms must address various constraints such as limited built-in gate sets, qubit coherence time, noise conditions, etc. Consequently, efficient transformation of quantum algorithms into executable quantum programs remains a core challenge in demonstrating quantum supremacy. The past decade, particularly with advancements in artificial intelligence, has witnessed significant progress in quantum logic circuit design and compilation optimization. Existing AI algorithms address each sub-step of the aforementioned processes separately. However, when the entire process is divided into multiple sub-modules processed by different algorithms, achieving unified goals between modules often results in suboptimal overall performance.

To maximize the potential and value of NISQ-era quantum processors and explore truly meaningful applications with quantum supremacy (distinct from random sampling tasks specifically designed for quantum computers [12, 197, 211]), more intelligent, efficient, and accurate logic circuit design and compilation optimization methods are required. Specifically, we propose that relevant methods should meet three main criteria: First, synthesize optimal logic circuits corresponding to quantum algorithms based on the quantum processors' built-in gate set. Given the current lack of a unified execution standard for quantum processors, algorithms must adapt to diverse built-in gate sets when synthesizing quantum logic circuits. Second, conduct qubit and coupler selection, mapping, and routing based on quantum processor calibration data. Algorithms should make real-time decisions based on dynamic qubit quality. Finally, balance theoretical accuracy and noise. Achieve optimal execution results on quantum processors while judiciously sacrificing theoretical accuracy. For instance, reducing circuit depth may decrease theoretical accuracy but could lead to better practical results by reducing circuit noise and minimizing the error between actual results and theoretical accuracy during execution.

We observe that optimization goals of individual sub-modules may not always align with the overall objective. Therefore, exploring end-to-end circuit synthesis optimization algorithms from quantum al-

gorithms to executable quantum programs is a key future direction. Leveraging artificial intelligence's powerful perception and decision-making capabilities, algorithms can consistently optimize for the overall goal of obtaining optimal practical results on quantum processors throughout the entire logic circuit design and compilation optimization process.

Currently, quantum computing resources are primarily deployed as cloud-based quantum platforms [79, 82]. Designing and deploying an end-to-end algorithm on a quantum cloud platform would greatly facilitate the exploration and development of practically valuable quantum algorithms. Users would only need to submit their designed quantum algorithms, and the artificial intelligence algorithm could obtain optimal experimental results based on real-time calibrated data. This approach would enhance algorithm performance and efficiency, reduce additional human costs following quantum algorithm design, and provide a new paradigm for quantum computing implementation. We anticipate increased engagement from the quantum computing community, particularly in quantum logic circuit design and compilation optimization, to collectively explore the vast potential of quantum computing.

References

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

2   Nabila Abdessaied, Mathias Soeken, and Rolf Drechsler. Quantum circuit optimization by hadamard gate reduction. In *Reversible Computation: 6th International Conference, RC 2014, Kyoto, Japan, July 10-11, 2014. Proceedings 6*, pages 149–162. Springer, 2014.

3   Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. Circuit compilation methodologies for quantum approximate optimization algorithm. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 215–228. IEEE, 2020.

4   Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *arXiv preprint quant-ph/0504218*, 2005.

5   Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

6   Mali2022optimizationtthew Amy, Parsiad Azimzadeh, and Michele Mosca. On the controlled-not complexity of controlled-not–phase circuits. *Quantum Science and Technology*, 4(1):015002, 2018.

7   Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time t-depth optimization of clifford+ t circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1476–1489, 2014.

8   Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

9   Matthew Amy and Michele Mosca. T-count optimization and reed–muller codes. *IEEE Transactions on Information Theory*, 65(8):4771–4784, 2019.

10 Panagiotis G Anastasiou, Yanzhu Chen, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. Tetris-adapt-vqe: An adaptive algorithm that yields shallower, denser circuit ansätze. *Physical Review Research*, 6(1):013254, 2024.

11 Lis Arufe, Miguel A González, Angelo Oddi, Riccardo Rasconi, and Ramiro Varela. Quantum circuit compilation by genetic algorithm for quantum approximate optimization algorithm applied to maxcut problem. *Swarm and Evolutionary Computation*, 69:101030, 2022.

12 Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

13 Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. Qure: Qubit re-allocation in noisy intermediate-scale quantum computers. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

14 J-H Bae, Paul M Alsing, Doyeol Ahn, and Warner A Miller. Quantum circuit optimization using quantum karnaugh map. *Scientific reports*, 10(1):15651, 2020.

15 Jeongho Bang and Seokwon Yoo. A genetic-algorithm-based method to find unitary transformations for any desired quantum computation and application to a one-bit oracle decision problem. *Journal of the Korean Physical Society*, 65(12):2001–2008, 2014.

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

17 Panagiotis Kl Barkoutsos, Jerome F Gonthier, Igor Sokolov, Nikolaj Moll, Gian Salis, Andreas Fuhrer, Marc Ganzhorn, Daniel J Egger, Matthias Troyer, Antonio Mezzacapo, et al. Quantum algorithms for electronic structure calculations: Particle-hole hamiltonian and optimized wave-function expansions. *Physical Review A*, 98(2):022322, 2018.

18 Johannes Bausch. Recurrent quantum neural networks. *Advances in neural information processing systems*, 33:1368–1379, 2020.

19 Collin Beaudoin, Koustubh Phalak, and Swaroop Ghosh. Altgraph: Redesigning quantum circuits using generative graph models for efficient optimization. *arXiv preprint arXiv:2403.12979*, 2024.

20 Y. Bengio, N. Leonard, and A.C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

21 FA Berazin. *The method of second quantization*, volume 24. Elsevier, 2012.

22 Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

23 Subodh Bijwe, Amit Kumar Chauhan, and Somitra Kumar Sanadhya. Implementing grover oracle for lightweight block ciphers under depth constraints. In *Australasian Conference on Information Security and Privacy*, pages 85–105. Springer, 2022.

24 Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66, 2024.

25 Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005.

26 Sergey Bravyi, Joseph A Latone, and Dmitri Maslov. 6-qubit optimal clifford circuits. *npj Quantum Information*, 8(1):79, 2022.

27 Sergey Bravyi, Ruslan Shaydulin, Shaohan Hu, and Dmitri Maslov. Clifford circuit optimization with templates and symbolic pauli gates. *Quantum*, 5:580, 2021.

28 Hugh GA Burton, Daniel Marti-Dafcik, David P Tew, and David J Wales. Exact electronic states with shallow quantum circuits from global optimisation. *npj Quantum Information*, 9(1):75, 2023.

29 Weizhou Cai, Yuwei Ma, Weiting Wang, Chang-Ling Zou, and Luyan Sun. Bosonic quantum error correction codes in superconducting quantum circuits. *Fundamental Research*, 1(1):50–67, 2021.

30 Earl T Campbell, Barbara M Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, 2017.

31 Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

32 Pranav Chandarana, Narendra N Hegade, Koushik Paul, Francisco Albarrán-Arriagada, Enrique Solano, Adolfo Del Campo, and Xi Chen. Digitized-counterdiabatic quantum approximate optimization algorithm. *Physical Review Research*, 4(1):013141, 2022.

33 Hongxiang Chen, Michael Vasmer, Nikolas P Breuckmann, and Edward Grant. Machine learning logical gates for quantum

error correction. *arXiv preprint arXiv:1912.10063*, 2019.

34 Lam Chi-Chung, P Sadayappan, and Rephael Wenger. On optimizing a class of multi-dimensional loops with reduction for parallel execution. *Parallel Processing Letters*, 7(02):157–168, 1997.

35 D Chivilikhin, A Samarin, V Ulyantsev, I Iorsh, AR Oganov, and O Kyriienko. Mog-vqe: Multiobjective genetic variational quantum eigensolver. *arXiv preprint arXiv:2007.04424*, 2020.

36 Bob Coecke and Ross Duncan. Interacting quantum observables. In *International Colloquium on Automata, Languages, and Programming*, pages 298–310. Springer, 2008.

37 Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

38 Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the qubit routing problem. *arXiv preprint arXiv:1902.08091*, 2019.

39 George Cybenko. Reducing quantum computations to elementary unitary operations. *Computing in science & engineering*, 3(2):27–32, 2001.

40 Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 291–303, 2019.

41 Marc G Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. Towards optimal topology aware quantum circuit synthesis. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 223–234. IEEE, 2020.

42 Marc Grau Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. Heuristics for quantum compiling with a continuous gate set. *arXiv preprint arXiv:1912.02727*, 2019.

43 Christopher M Dawson and Michael A Nielsen. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030*, 2005.

44 Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to reduce $\pi/4$-parity-phase circuits, motivated by the ZX calculus. *Electronic Proceedings in Theoretical Computer Science*, 318:131–149, may 2020.

45 Vitaly G Deibuk and Andrij V Biloshytskyi. Design of a ternary reversible/quantum adder using genetic algorithm. *International Journal of Information Technology and Computer Science*, 7(9):38–45, 2015.

46 David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

47 Shengchao Ding, Zhi Jin, and Qing Yang. Evolving quantum oracles with hybrid quantum-inspired evolutionary algorithm. *arXiv preprint quant-ph/0610105*, 2006.

48 Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Information*, 8(1):1–8, 2022.

49 Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum*, 4:279, jun 2020.

50 Trong Duong, Sang T Truong, Minh Tam, Bao Bach, Ju-Young Ryu, and June-Koo Kevin Rhee. Quantum neural architecture search with quantum circuits metric and bayesian optimization. *arXiv preprint arXiv:2206.14115*, 2022.

51 Andrew Fagan and Ross Duncan. Optimising clifford circuits with quantomatic. *Electronic Proceedings in Theoretical Computer Science*, 287:85–105, jan 2019.

52 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

53 Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

54 Dmitry A Fedorov, Bo Peng, Niranjan Govind, and Yuri Alexeev. Vqe method: a short survey and recent developments. *Materials Theory*, 6(1):2, 2022.

55 Simon Forest, David Gosset, Vadym Kliuchnikov, and David McKinnon. Exact synthesis of single-qubit unitaries over clifford-cyclotomic gate sets. *Journal of Mathematical Physics*, 56(8), 2015.

56 Thomas Fösel, Murphy Yuezhen Niu, Florian Marquardt, and Li Li. Quantum circuit optimization with deep reinforcement learning. *arXiv preprint arXiv:2103.07585*, 2021.

57 Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Physical Review A*, 80(5):052312, 2009.

58 Vlad Gheorghiu, Jiaxin Huang, Sarah Meng Li, Michele Mosca, and Priyanka Mukhopadhyay. Reducing the cnot count for clifford+ t circuits on nisq architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

59 Stefano Gogioso and Richie Yeung. Annealing optimisation of mixed zx phase circuits. *arXiv preprint arXiv:2206.11839*, 2022.

60 Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

61 Daniel Gottesman. The heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.

62 Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127, 1998.

63 Harper R Grimsley, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature communications*, 10(1):3007, 2019.

64 Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

65 Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

66 E.J. Gumbel. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 1954.

67 Reza Haghshenas, Johnnie Gray, Andrew C Potter, and Garnet Kin-Lic Chan. Variational power of quantum circuit tensor networks. *Physical Review X*, 12(1):011047, 2022.

68 Sonaldeep Halder, Anish Dey, Chinmay Shrikhande, and Rahul Maitra. Machine learning assisted construction of a shallow depth dynamic ansatz for noisy quantum hardware. *Chemical Science*, 2024.

69 Yong He, Ming-Xing Luo, E Zhang, Hong-Ke Wang, and Xiao-Feng Wang. Decompositions of n-qubit toffoli gates with linear circuit complexity. *International Journal of Theoretical Physics*, 56:2350–2361, 2017.

70 Zhimin He, Maijie Deng, Shenggen Zheng, Lvzhou Li, and Haozhen Situ. Gsqas: graph self-supervised quantum architecture search. *Physica A: Statistical Mechanics and its Applications*, 630:129286, 2023.

71 Zhimin He, Maijie Deng, Shenggen Zheng, Lvzhou Li, and Haozhen Situ. Training-free quantum architecture search. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 12430–12438, 2024.

72 Zhimin He, Xuefen Zhang, Chuangtao Chen, Zhiming Huang, Yan Zhou, and Haozhen Situ. A gnn-based predictor for quantum architecture search. *Quantum Information Processing*, 22(2):128, 2023.

73 Rebekah Herrman, Phillip C Lotshaw, James Ostrowski, Travis S Humble, and George Siopsis. Multi-angle quantum approximate optimization algorithm. *Scientific Reports*, 12(1):6781, 2022.

74 Rebekah Herrman, Lorna Treffert, James Ostrowski, Phillip C Lotshaw, Travis S Humble, and George Siopsis. Globally optimizing qaoa circuit depth for constrained optimization problems. *Algorithms*, 14(10):294, 2021.

75 Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces t count. *Quantum Science and Technology*, 4(1):015004, 2018.

76 Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu, and Michael Hicks. A verified optimizer for quantum circuits. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–29, 2021.

77 Ching-Yao Huang, Chi-Hsiang Lien, and Wai-Kei Mak. Reinforcement learning and dear framework for solving the qubit mapping problem. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

78 Cupjin Huang, Fang Zhang, Michael Newman, Xiaotong Ni, Dawei Ding, Junjie Cai, Xun Gao, Tenghui Wang, Feng Wu, Gengyan Zhang, et al. Efficient parallelization of tensor network contraction for simulating quantum computation. *Nature Computational Science*, 1(9):578–587, 2021.

79 He-Liang Huang, Ashutosh K Goswami, Wan-Su Bao, and Prasanta K Panigrahi. Demonstration of essentiality of entanglement in a deutsch-like quantum algorithm. *Science China Physics, Mechanics & Astronomy*, 61:1–7, 2018.

80 He-Liang Huang, Dachao Wu, Daojin Fan, and Xiaobo Zhu. Superconducting quantum computing: a review. *Science China Information Sciences*, 63(8):1–32, 2020.

81 He-Liang Huang, Xiao-Yue Xu, Chu Guo, Guojing Tian, Shi-Jie Wei, Xiaoming Sun, Wan-Su Bao, and Gui-Lu Long. Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation. *Science China Physics, Mechanics & Astronomy*, 66(5):250302, 2023.

82 He-Liang Huang, You-Wei Zhao, Tan Li, Feng-Guang Li, Yu-Tao Du, Xiang-Qun Fu, Shuo Zhang, Xiang Wang, and Wan-Su Bao. Homomorphic encryption experiments on ibm's cloud quantum computing platform. *Frontiers of Physics*, 12:1–6, 2017.

83 Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 2022.

84 William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.

85 Raban Iten, Romain Moyard, Tony Metger, David Sutter, and Stefan Woerner. Exact and practical pattern matching for quantum circuit optimization. *ACM Transactions on Quantum Computing*, 3(1):1–41, 2022.

86 E Jang, S Gu, and B Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

87 Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing grover oracles for quantum key search on aes and lowmc. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*, pages 280–310. Springer, 2020.

88 Jiaqing Jiang, Xiaoming Sun, Shang-Hua Teng, Bujiao Wu, Kewen Wu, and Jialin Zhang. Optimal space-depth trade-off of cnot circuits in quantum logic synthesis. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 213–229. SIAM, 2020.

89 Pascual Jordan and Eugene Paul Wigner. *Über das paulische äquivalenzverbot*. Springer, 1993.

90 Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *nature*, 549(7671):242–246, 2017.

91 Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T Sornborger, and Patrick J Coles. Quantum-assisted quantum compiling. *Quantum*, 3:140, 2019.

92 Aleks Kissinger and Arianne Meijer-van de Griend. Cnot circuit extraction for topologically-constrained quantum memories. *arXiv preprint arXiv:1904.00633*, 2019.

93 Aleks Kissinger and John van de Wetering. PyZX: Large scale automated diagrammatic reasoning. *Electronic Proceedings in Theoretical Computer Science*, 318:229–241, may 2020.

94 Aleks Kissinger and John van de Wetering. Reducing the number of non-clifford gates in quantum circuits. *Physical Review A*, 102(2), aug 2020.

95 Aleks Kissinger and John van de Wetering. Reducing the number of non-clifford gates in quantum circuits. *Physical Review A*, 102(2):022406, 2020.

96 Aleks Kissinger and Vladimir Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In *International Conference on Automated Deduction*, pages 326–336. Springer, 2015.

97 Alexei Yu Kitaev, Alexander Shen, and Mikhail N Vyalyi. *Classical and quantum computation*. Number 47. American Mathematical Soc., 2002.

98 Ian D Kivlichan, Jarrod McClean, Nathan Wiebe, Craig Gidney, Alán Aspuru-Guzik, Garnet Kin-Lic Chan, and Ryan Babbush. Quantum simulation of electronic structure with linear depth and connectivity. *Physical review letters*, 120(11):110501, 2018.

99 Vadym Kliuchnikov and Dmitri Maslov. Optimization of clifford circuits. *Physical Review A*, 88(5):052307, 2013.

100 Emanuel Knill. Approximation by quantum circuits. *arXiv preprint quant-ph/9508006*, 1995.

101 En-Jui Kuo, Yao-Lung L Fang, and Samuel Yen-Chi Chen. Quantum architecture search via deep reinforcement learning. *arXiv preprint arXiv:2104.07715*, 2021.

102 Janusz Kusyk, Samah M Saeed, and Muharrem Umit Uyar. Survey on quantum circuit compilation for noisy intermediate-scale quantum computers: Artificial intelligence to heuristics. *IEEE Transactions on Quantum Engineering*, 2:1–16, 2021.

103 Lucas Lamata, Unai Alvarez-Rodriguez, José David Martín-Guerrero, Mikel Sanz, and Enrique Solano. Quantum autoencoders via quantum adders with genetic algorithms. *Quantum Science and Technology*, 4(1):014007, 2018.

104 Urtzi Las Heras, Unai Alvarez-Rodriguez, Enrique Solano, and Mikel Sanz. Genetic algorithms for digital quantum simulations. *Physical review letters*, 116(23):230504, 2016.

105 Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

106   Daniel D Lee, P Pham, Y Largman, and A Ng. Advances in neural information processing systems 22. *Tech Rep*, 2009.

107   Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019.

108   Rui Li, Unai Alvarez-Rodriguez, Lucas Lamata, and Enrique Solano. Approximate quantum adders with genetic algorithms: an ibm quantum experience. *Quantum Measurements and Quantum Metrology*, 4(1):1–7, 2017.

109   Sanjiang Li, Xiangzhen Zhou, and Yuan Feng. Qubit mapping based on subgraph isomorphism and filtered depth-limited search. *IEEE Transactions on Computers*, 70(11):1777–1788, 2020.

110   Zikun Li, Jinjun Peng, Yixuan Mei, Sina Lin, Yi Wu, Oded Padon, and Zhihao Jia. Quarl: A learning-based quantum circuit optimizer. *arXiv preprint arXiv:2307.10120*, 2023.

111   Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

112   Ji Liu, Luciano Bello, and Huiyang Zhou. Relaxed peephole optimization: A novel compiler optimization for quantum circuits. In *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 301–314. IEEE, 2021.

113   Ji Liu, Peiyi Li, and Huiyang Zhou. Not all swaps have the same cost: A case for optimization-aware qubit routing. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 709–725. IEEE, 2022.

114   Lei Liu and Xinglei Dou. Qucloud: A new qubit mapping mechanism for multi-programming quantum computing in cloud environment. In *2021 IEEE International symposium on high-performance computer architecture (HPCA)*, pages 167–178. IEEE, 2021.

115   Xudong Lu, Kaisen Pan, Ge Yan, Jiaming Shan, Wenjie Wu, and Junchi Yan. QAS-bench: Rethinking quantum architecture search and a benchmark. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22880–22898. PMLR, 23–29 Jul 2023.

116   Dmitry I Lyakh. An efficient tensor transpose algorithm for multicore cpu, intel xeon phi, and nvidia tesla gpu. *Computer Physics Communications*, 189:84–91, 2015.

117   Ilias Magoulas and Francesco A Evangelista. Linear-scaling quantum circuits for computational chemistry. *Journal of Chemical Theory and Computation*, 19(15):4815–4821, 2023.

118   Ritajit Majumdar, Debasmita Bhoumik, Dhiraj Madan, Dhinakaran Vinayagamurthy, Shesha Raghunathan, and Susmita Sur-Kolay. Depth optimized ansatz circuit in qaoa for max-cut. *arXiv preprint arXiv:2110.04637*, 2021.

119   Ritajit Majumdar, Dhiraj Madan, Debasmita Bhoumik, Dhinakaran Vinayagamurthy, Shesha Raghunathan, and Susmita Sur-Kolay. Optimizing ansatz design in qaoa for max-cut. *arXiv preprint arXiv:2106.02812*, 2021.

120   Igor L Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.

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

122   Dmitri Maslov, Gerhard W Dueck, and D Michael Miller. Simplification of toffoli networks via templates. In *16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings.*, pages 53–58. IEEE, 2003.

123   Dmitri Maslov, Gerhard W Dueck, D Michael Miller, and Camille Negrevergne. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):436–444, 2008.

124   Dmitri Maslov, Christina Young, D Michael Miller, and Gerhard W Dueck. Quantum circuit simplification using templates. In *Design, Automation and Test in Europe*, pages 1208–1213. IEEE, 2005.

125   Paul Massey, John A Clark, and Susan Stepney. Evolving quantum circuits and programs through genetic programming. In *Genetic and Evolutionary Computation Conference*, pages 569–580. Springer, 2004.

126   Yuta Matsuzawa and Yuki Kurashige. Jastrow-type decomposition in quantum chemistry for low-depth quantum circuits. *Journal of chemical theory and computation*, 16(2):944–952, 2020.

127   William M McKeeman. Peephole optimization. *Communications of the ACM*, 8(7):443–444, 1965.

128   Giulia Meuli, Mathias Soeken, and Giovanni De Micheli. Sat-based {CNOT, T} quantum circuit synthesis. In *Reversible Computation: 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings 10*, pages 175–188. Springer, 2018.

129   D Michael Miller, Robert Wille, and Zahra Sasanian. Elementary quantum gate realizations for multiple-control toffoli gates. In *2011 41st IEEE international symposium on multiple-valued logic*, pages 288–293. IEEE, 2011.

130   Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. Qubit mapping and routing via maxsat. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1078–1091. IEEE, 2022.

131   Mario Motta, Erika Ye, Jarrod R McClean, Zhendong Li, Austin J Minnich, Ryan Babbush, and Garnet Kin-Lic Chan. Low rank representations for quantum simulation of electronic structure. *npj Quantum Information*, 7(1):83, 2021.

132   Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Quantum circuits for general multiqubit gates. *Physical review letters*, 93(13):130502, 2004.

133   Mikko Möttönen[1] and Juha J Vartiainen. Decompositions of general quantum gates. *Trends in quantum computing research*, page 149, 2006.

134   Anthony Munson, Bob Coecke, and Quanlong Wang. AND-gates in ZX-calculus: Spider nest identities and QBC-completeness. *Electronic Proceedings in Theoretical Computer Science*, 340:230–255, sep 2021.

135   Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pages 1015–1029, 2019.

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

137   Yunseong Nam, Yuan Su, and Dmitri Maslov. Approximate quantum fourier transform with o (n log (n)) t gates. *NPJ Quantum Information*, 6(1):26, 2020.

138   Giacomo Nannicini, Lev S Bishop, Oktay Günlük, and Petar Jurcevic. Optimal qubit assignment and routing via integer programming. *ACM Transactions on Quantum Computing*, 4(1):1–31, 2022.

139   Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. Quantum circuit optimizations for nisq architectures. *Quantum Science and Technology*, 5(2):025010, 2020.

140   Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J Briegel, and Nicolai Friis. Optimizing quantum error

correction codes with reinforcement learning. *Quantum*, 3:215, 2019.

141 Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

142 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010.

143 Philipp Niemann, Robert Wille, David Michael Miller, Mitchell A Thornton, and Rolf Drechsler. Qmdds: Efficient quantum function representation and manipulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):86–99, 2015.

144 Siyuan Niu, Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. A hardware-aware heuristic for the qubit mapping problem in the nisq era. *IEEE Transactions on Quantum Engineering*, 1:1–14, 2020.

145 Siyuan Niu and Aida Todri-Sanial. Enabling multi-programming mechanism for quantum computing in the nisq era. *Quantum*, 7:925, 2023.

146 Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 349:117–158, 2014.

147 Mateusz Ostaszewski, Lea M Trenkwalder, Wojciech Masarczyk, Eleanor Scerri, and Vedran Dunjko. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34:18182–18194, 2021.

148 Sunghye Park, Daeyeon Kim, Minhyuk Kweon, Jae-Yoon Sim, and Seokhyeong Kang. A fast and scalable qubit-mapping method for noisy intermediate-scale quantum computers. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 13–18, 2022.

149 Tirthak Patel, Ed Younis, Costin Iancu, Wibe de Jong, and Devesh Tiwari. Robust and resource-efficient quantum circuit approximation. *arXiv preprint arXiv:2108.12714*, 2021.

150 Yash J Patel, Akash Kundu, Mateusz Ostaszewski, Xavier Bonet-Monroig, Vedran Dunjko, and Onur Danaci. Curriculum reinforcement learning for quantum architecture search under hardware errors. *arXiv preprint arXiv:2402.03500*, 2024.

151 Roger Penrose. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1:221–244, 1971.

152 Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

153 Jessica Pointing, Oded Padon, Zhihao Jia, Henry Ma, Auguste Hirth, Jens Palsberg, and Alex Aiken. Quanto: Optimizing quantum circuits with automatic generation of circuit identities. *arXiv preprint arXiv:2111.11387*, 2021.

154 Václav Potoček, Alan P Reynolds, Alessandro Fedrizzi, and David W Corne. Multi-objective evolutionary algorithms for quantum circuit discovery. *arXiv preprint arXiv:1812.04458*, 2018.

155 Matteo G Pozzi, Steven J Herbert, Akash Sengupta, and Robert D Mullins. Using reinforcement learning to perform qubit routing in quantum compilers. *ACM Transactions on Quantum Computing*, 3(2):1–25, 2022.

156 Aditya K Prasad, Vivek V Shende, Igor L Markov, John P Hayes, and Ketan N Patel. Data structures and algorithms for simplifying reversible circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2(4):277–293, 2006.

157 John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

158 Mostafizar Rahman and Goutam Paul. Grover on katan: Quantum resource estimation. *IEEE Transactions on Quantum Engineering*, 3:1–9, 2022.

159 Arthur G Rattew and Patrick Rebentrost. Non-linear transformations of quantum amplitudes: Exponential improvement, generalization, and applications. *arXiv preprint arXiv:2309.09839*, 2023.

160 Alex Rigby. *Heuristics in quantum error correction*. PhD thesis, University of Tasmania, 2021.

161 Jordi Riu, Jan Nogué, Gerard Vilaplana, Artur Garcia-Saez, and Marta P Estarellas. Reinforcement learning based quantum circuit optimization via zx-calculus. *arXiv preprint arXiv:2312.11597*, 2023.

162 J Romero, R Babbush, JR McClean, C Hempel, P Love, and A Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *arXiv preprint arXiv:1701.02691*, 2017.

163 Francisco JR Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco JH Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering, et al. Quantum circuit optimization with alphatensor. *arXiv preprint arXiv:2402.14396*, 2024.

164 Alberto Sanfeliu and Manuel Lazo Cortés. Progress in pattern recognition, image analysis and applications. In *9th Iberoamerican Congress on Pattern Recognition, CIARP*. Springer, 2004.

165 Mariia D Sapova and Aleksey K Fedorov. Variational quantum eigensolver techniques for simulating carbon monoxide oxidation. *Communications Physics*, 5(1):1–13, 2022.

166 Sarah Schneider, Lukas Burgholzer, and Robert Wille. A sat encoding for optimal clifford circuit synthesis. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 190–195, 2023.

167 Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13:2567–2586, 2014.

168 Roman Schutski, Taras Khakhulin, Ivan Oseledets, and Dmitry Kolmakov. Simple heuristics for efficient parallel tensor contraction and quantum circuit simulation. *Physical Review A*, 102(6):062614, 2020.

169 Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum logic circuits. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 272–275, 2005.

170 Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

171 Animesh Sinha, Utkarsh Azad, and Harjinder Singh. Qubit routing using graph neural network aided monte carlo tree search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9935–9943, 2022.

172 Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, pages 113–125, 2018.

173 Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t— ket¿: a retargetable compiler for nisq devices. *Quantum Science and Technology*, 6(1):014003, 2020.

174 Stasja Stanisic, Jan Lukas Bosse, Filippo Maria Gambetta, Raul A Santos, Wojciech Mruczkiewicz, Thomas E O'Brien, Eric Ostby, and Ashley Montanaro. Observing ground-state properties of the fermi-hubbard model using a scalable algorithm on a quantum computer. *Nature communications*, 13(1):5743, 2022.

175   Korbinian Staudacher. *Optimization Approaches for Quantum Circuits using ZX-calculus.* PhD thesis, Master's thesis, Ludwig-Maximilians-Universität, München. Available at https . . . , 2021.

176   Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann, and Wolfgang Gehrke. Reducing 2-qubit gate count for zx-calculus based quantum circuit optimization. *arXiv preprint arXiv:2311.08881*, 2023.

177   Ho Lun Tang, VO Shkolnikov, George S Barron, Harper R Grimsley, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum*, 2(2):020310, 2021.

178   Swamit S Tannu and Moinuddin 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*, pages 987–999, 2019.

179   Andrew G Taube and Rodney J Bartlett. New perspectives on unitary coupled-cluster theory. *International journal of quantum chemistry*, 106(15):3393–3401, 2006.

180   Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

181   Nikolay V Tkachenko, James Sud, Yu Zhang, Sergei Tretiak, Petr M Anisimov, Andrew T Arrasmith, Patrick J Coles, Lukasz Cincio, and Pavel A Dub. Correlation-informed permutation of qubits for reducing ansatz depth in the variational quantum eigensolver. *PRX Quantum*, 2(2):020337, 2021.

182   John van de Wetering. Constructing quantum circuits with global gates. *New Journal of Physics*, 23(4):043015, apr 2021.

183   Vivien Vandaele, Simon Martiel, and Timothée Goubault de Brugière. Phase polynomials synthesis algorithms for nisq architectures and beyond. *Quantum Science and Technology*, 7(4):045027, 2022.

184   Juha J Vartiainen, Mikko Möttönen, and Martti M Salomaa. Efficient decomposition of quantum gates. *Physical review letters*, 92(17):177902, 2004.

185   Trevor Vincent, Lee J O'Riordan, Mikhail Andrenkov, Jack Brown, Nathan Killoran, Haoyu Qi, and Ish Dhand. Jet: Fast quantum circuit simulations with parallel task-based tensor-network contraction. *Quantum*, 6:709, 2022.

186   Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Diskret analiz*, 3:25–30, 1964.

187   Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 692–708. IEEE, 2022.

188   X. Wang, J. Lin, J. Zhao, X. Yang, and J. Yan. Eautodet: Efficient architecture search for object detection. In *European Conference on Computer Vision*, 2022.

189   Yaakov S Weinstein, MA Pravia, EM Fortunato, Seth Lloyd, and David G Cory. Implementation of the quantum fourier transform. *Physical review letters*, 86(9):1889, 2001.

190   Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

191   Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations. in 2019 56th acm/ieee design automation conference (dac). *arXiv preprint arxiv:1907.02026*, 2019.

192   Colin P Williams and Alexander G Gray. Automated design of quantum circuits. In *NASA International Conference on Quantum Computing and Quantum Communications*, pages 113–125. Springer, 1999.

193   David Winderl. Zx polynomial synthesis. 2022.

194   Wenjie Wu, Yiquan Wang, Ge Yan, Yuming Zhao, and Junchi Yan. On reducing the execution latency of superconducting quantum processors via quantum program scheduling. *arXiv preprint arXiv:2404.07882*, 2024.

195   Wenjie Wu, Ge Yan, Xudong Lu, Kaisen Pan, and Junchi Yan. Quantumdarts: differentiable quantum architecture search for variational quantum algorithms. In *International Conference on Machine Learning*, pages 37745–37764. PMLR, 2023.

196   Xin-Chuan Wu, Marc Grau Davis, Frederic T Chong, and Costin Iancu. Qgo: Scalable quantum circuit optimization using automated synthesis. *arXiv preprint arXiv:2012.09835*, 2020.

197   Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, et al. Strong quantum computational advantage using a superconducting quantum processor. *Physical review letters*, 127(18):180501, 2021.

198   Amanda Xu, Abtin Molavi, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. Synthesizing quantum-circuit optimizers. *Proceedings of the ACM on Programming Languages*, 7(PLDI):835–859, 2023.

199   Mingkuan Xu, Zikun Li, Oded Padon, Sina Lin, Jessica Pointing, Auguste Hirth, Henry Ma, Jens Palsberg, Alex Aiken, Umut A Acar, et al. Quartz: superoptimization of quantum circuits. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 625–640, 2022.

200   Nikola Yanakiev, Normann Mertig, Christopher K Long, and David RM Arvidsson-Shukur. Dynamic adaptive quantum approximate optimization algorithm for shallow, noise-resilient circuits. *Physical Review A*, 109(3):032420, 2024.

201   Yong-Xin Yao, Niladri Gomes, Feng Zhang, Cai-Zhuang Wang, Kai-Ming Ho, Thomas Iadecola, and Peter P Orth. Adaptive variational quantum dynamics simulations. *PRX Quantum*, 2(3):030307, 2021.

202   Esther Ye and Samuel Yen-Chi Chen. Quantum architecture search via continual reinforcement learning. *arXiv preprint arXiv:2112.05779*, 2021.

203   Richie Yeung. Diagrammatic design and study of ansätze for quantum machine learning. *arXiv preprint arXiv:2011.11073*, 2020.

204   Yordan S Yordanov, Vasileios Armaos, Crispin HW Barnes, and David RM Arvidsson-Shukur. Qubit-excitation-based adaptive variational quantum eigensolver. *Communications Physics*, 4(1):228, 2021.

205   Yexiong Zeng, Zheng-Yang Zhou, Enrico Rinaldi, Clemens Gneiting, and Franco Nori. Approximate autonomous quantum error correction with reinforcement learning. *arXiv preprint arXiv:2212.11651*, 2022.

206   Chi Zhang, Ari B Hayes, Longfei Qiu, Yuwei Jin, Yanhao Chen, and Eddy Z Zhang. Time-optimal qubit mapping. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 360–374, 2021.

207   Fang Zhang and Jianxin Chen. Optimizing t gates in clifford+ t circuit as $\frac{\pi}{4}$ rotations around paulis. *arXiv preprint arXiv:1903.12456*, 2019.

208   Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Neural predictor based quantum architecture search. *Machine Learning: Science and Technology*, 2(4):045027, 2021.

209   Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Differentiable quantum architecture search. *Quantum*

Science and Technology, 7(4):045023, 2022.

210 Zi-Jian Zhang, Jinzhao Sun, Xiao Yuan, and Man-Hong Yung. Low-depth hamiltonian simulation by an adaptive product formula. *Physical Review Letters*, 130(4):040601, 2023.

211 Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

212 Xiangzhen Zhou, Yuan Feng, and Sanjiang Li. A monte carlo tree search framework for quantum circuit transformation. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–7, 2020.

213 Xiangzhen Zhou, Sanjiang Li, and Yuan Feng. Quantum circuit transformation based on simulated annealing and heuristic search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4683–4694, 2020.

214 Xinlan Zhou, Debbie W Leung, and Isaac L Chuang. Methodology for quantum logic gate construction. *Physical Review A*, 62(5):052316, 2000.

215 Linghua Zhu, Ho Lun Tang, George S Barron, FA Calderon-Vargas, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer. *Physical Review Research*, 4(3):033029, 2022.

216 Pengcheng Zhu, Zhijin Guan, and Xueyun Cheng. A dynamic look-ahead heuristic for the qubit mapping problem of nisq computers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4721–4735, 2020.

217 Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the ibm qx architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(7):1226–1236, 2018.