Compiling Local Hamiltonians

Seyed Sajad Kahani
22222815
Supervisor: Prof. Dan Browne

June 11, 2023

Contents

Abstract

This is a review essay that discusses the importance of local Hamiltonians in NISQ algorithms and the value of a compiler for them. The essay covers the basic concepts relevant to Hamiltonian compilation, including the Suzuki-Trotter formula, and examines the state-of-the-art methods in general quantum compilation. A structural approach is taken to explore the problems and algorithms associated with local Hamiltonian compilation. Finally, the essay discusses the open problems that will be addressed in the project, including possible ideas for extending gate set as well as improving scheduling.

Chapter 1

Introduction

Quantum computation is an emerging field that aims to use quantum mechanics to solve problems that are intractable for classical computers. Since the earliest conceptualization of quantum computation [14], it has been believed that quantum computers could revolutionize the way we solve problems, particularly those involving simulating nature. Over time, it has become clear that quantum computers have applications far beyond physical simulations. There are algorithms for search and traversing graphs, solving linear equations [24], and methods for machine learning and optimization [19].

Despite significant efforts, we are still far from fully utilizing these algorithms. Our current hardware technology has not yet achieved the desired accuracy and number of qubits necessary for quantum computers to outperform classical computers in solving useful problems. This situation is commonly referred to as the "noisy intermediate scale quantum" (NISQ) era. In this era, the main barriers to run a general quantum algorithm are the effects of noise (especially for entangling gates) and decoherence time, which limits the depth of the circuit [29].

Nevertheless, it is believed that some algorithms can still produce valuable results on NISQ devices. Quantum Variational Algorithms [6], as well as some simulation algorithms, are promising candidates for this purpose [29, 21]. Therefore, a problem-specific approach to quantum computation would be beneficial in the short-term.

Even for these promising candidates, circuit optimization is necessary [9, 13]. This raises the need for a specific-purpose quantum compiler. Moreover, while local Hamiltonians are commonly used in NISQ-era algorithms[12], a compiler for them would be a valuable tool.

This project aims to improve existing methods for compiling local Hamiltonians, and this essay will focus on the literature review of these existing methods.

Various topics ranging from classical compilation to quantum information theory are not directly related to quantum compilers but are still beneficial in this study. These topics will be covered in chapter 2. In chapter 3, we will review the literature on quantum compilers, as well as the state-of-the-art methods in the quantum compilation of local Hamiltonians. Chapter 4 will discuss the open problems that will be addressed in the project itself. Finally, we will conclude the essay in chapter 6.

Chapter 2

Basic Concepts

2.1 Classical Compilation

A detailed study of the principles and heritage of classical compilers could be insightful and influential for any research conducted in the quantum compilation area. Rather than focusing on state-of-the-art results in classical compilation, we will focus on well-established principles to share the experience of classical compilers with quantum compilers.

In the context of executing computer programs, there are two main approaches: interpretation and compilation. Interpretation is the process of processing a program at the time of execution. It means that the processor of the program is aware of the input and the execution environment as well. While compilation refers to the process of transforming a program into another representation before the execution time, so the input and environment are not known at the moment [1, p. 2].

For the quantum case, the transformation of programs (that are represented as gates) will be done before the execution, and the situation is quite similar to classical compilation. We can also assume the possibility of quantum interpretation, which is beyond our scope [16].

The process of compilation can be divided into three main phases:

- **Parsing**: Using lexical and syntactic analysis, the source code is transformed into an abstract syntax tree (AST).
- **Intermediate Code Generation**: The AST is transformed into an intermediate representation (IR), which is a lower-level representation that is still device-independent. Any device-independent optimization will also take place here.
- **Target Code Generation**: The IR is transformed into the target code, which is the final representation of the program. The target code is device-dependent, and it is the final representation of the program.

Because of the common representation of quantum programs, a quantum compiler will never tackle the parsing phase, while the next two phases could be useful as a guideline for a quantum compiler.

Yet, the influence of classical compilers on quantum compilers may not be limited to the breakdown structure of the compilation process. They will also share some similar subproblems that we define as follows.

Problem 1 (Register Allocation). *Registers are the fastest memory in a computer. However, they are limited in number and size. Therefore, the compiler must decide at each time which variables should be stored in registers and which should be stored in memory. [2, pp. 440-444]*

Problem 2 (Instruction Scheduling). *There are some degrees of freedom regarding the order of instructions in a program. The compiler must decide which order is best for the program. [2, chap. 10]*

Problem 3 (Code Motion). *Code motion is the process of moving code to a different location in the program. This process may result in changes in the code snippets that are moved, but it will not change the semantics of the whole program. The compiler can move code to different locations to improve the performance of the program. [1, p. 592]*

2.2 Quantum Computation

Quantum computation could be defined using different models, such as quantum lambda calculus or quantum Turing machines. However, the most common model is the quantum circuit model, which is a model that is based on the quantum circuit model of quantum mechanics. In this model, a quantum computer is a quantum system that is prepared in a specific state, namely a state of n qubits, that are typically in a basic product state. Then a sequence of quantum gates could be applied on any subset of qubits. By applying a quantum gate localy, we mean applying the indentity on the rest of the qubits. Finally, the state of the qubits is measured, and the result is the output of the quantum computer.

Now we will study a few more tools that are useful for our study.

One set of these tools is Decompositions of unitaries. They could be considered as a form of compilations as they transform a gate into a sequence of gates that are often simpler.

2.3 Suzuki-Trotter Decomposition

The Suzuki-Trotter decomposition [36, 35] is one of the most important tools for dealing with Hamiltonians. This decomposition approximates the time evolution of a Hamiltonian with a sequence of time evolutions of simpler Hamiltonians (terms of the first Hamiltonian). The simplest case is called the Lie-Trotter formula, and it is stated as follows.

Theorem 1 (Lie Trotter Formula)**.** *For any $H = A + B$,*

$$e^{iH} = \lim_{n\to\infty} (e^{iA/n} e^{iB/n})^n. \tag{2.1}$$

Generally, the decomposition is stated as follows.

Theorem 2 (Higher Order Suzuki-Trotter Decomposition)**.** *For any $H = \sum_{j=1}^{k} h_j$, we can define a series of approximations called $\tilde{U}_{2p}$ that*

$$\tilde{U}_1(t) = e^{ih_1 t} \dots e^{ih_k t}, \tag{2.2}$$

$$\tilde{U}_2(t) = e^{ih_1 t/2} \dots e^{ih_k t/2} e^{ih_k t/2} \dots e^{ih_1 t/2}, \tag{2.3}$$

$$\tilde{U}_{2p}(t) = \tilde{U}_{2p-2}(u_p t)^2 \tilde{U}_{2p-2}((1-4u_p)t) \tilde{U}_{2p-2}(u_p t)^2, \tag{2.4}$$

where $u_p = 1/(4 - 4^{1/(2p-1)})$.

Then,

$$\left\| e^{iHt} - \tilde{U}_{2p}(t) \right\| \in O(t^{2p+1}). \tag{2.5}$$

This could be considered the baseline process for the compilation of Hamiltonians.

2.4 Kraus-Cirac Decomposition

The Kraus-Cirac decomposition [20] helps us create arbitrary two-qubit gates. It states that any two-qubit gate can be created by a Heisenberg model and local gates.

Theorem 3 (Kraus-Cirac Decomposition)**.** *For any $U \in SU(4)$, there exist $V_1, V_2, V_3, V_4 \in SU(2)$ together with $\alpha, \beta, \gamma \in \mathbb{R}$ such that*

$$U = V_1 \otimes V_2 e^{\alpha X\otimes X + \beta Y \otimes Y + \gamma Z \otimes Z} V_3 \otimes V_4. \tag{2.6}$$

This theorem, together with the basic properties of entanglement power, will also lead to many more results on communication using two-qubit gates [4] and also the universality and optimality of three CNOTs for two-qubit gates [37].

2.5 Entanglement Power

We already know that there are so many measures for the entanglement of a bipartite state, building upon those measures, we can define entanglement power.

Entanglement power is a measure of the ability of a two-qubit gate to create entanglement. It has multiple definitions, such as the maximum amount of ebits that can be created from product states [31], or the average amount of them (with respect to a Haar distribution) [38], or even the number of terms in a

Schmidt decomposition (which will be equal to the number of non-zero terms in the Kraus-Cirac decomposition) [27].

This measure, assigns a number to each gate, and then, by composing gates, the entanglement power could not exceed the summation of the entanglement powers of the gates that are used to create the gate. This fact is used to prove many tight bounds for decomposition of two-qubit gates.

Moreover, these efforts implicitly define a hierarchy of two-qubit gates based on the number of non-zero terms in the interaction. For example, CNOT has one non-zero term while SWAP has three.

Chapter 3

Related Works

To understand the current state of the art in quantum compilation, we will first look at the general compilation problem.

3.1 General Quantum Compilation

The term quantum compilation is used in the literature for any process that transforms a higher-level description of a quantum algorithm into a lower-level description. [17]

While this definition is very broad, currently the most of the research in this area is focused on the process of compiling a quantum circuit to a circuit that can be implemented on a device. This process is also called circuit transformation or circuit mapping.

Note that rather the general quantum compilation tasks which involves different representations of quantum algorithms, in the circuit mapping problem we are only dealing with the circuit representations (DAGs) that are defined as below.

Definition 1 (Circuit)**.** *A circuit is a set of operations (gates and measurements) on a defined set of qubits (or any other Hilbert space) that is associated with a DAG to represent dependencies between operations.*

Then, the process could be seen as a transformation between the more relaxed circuits into the circuits that are satisfying the constraints of the device. We can define the circuit and its imposed constraints as below.

Definition 2 (Device)**.** *A device is defined by a connectivity graph and a set of gates that can be used on the qubits. Optionally, it may include functions defining nodes or edges to represent errors.*

This process is often divided into subproblems that are selectively discussed here.

3.1.1 Qubit Allocation

The definition of qubit allocation may vary in the literature, but we can roughly define it as below

Problem 4 (Qubit Allocation)**.** *Qubit allocation is the problem of assigning physical qubits of a device to logical qubits of a quantum circuit at each time step to minimize the circuit's complexity.*

We can see that this problem is similar to problem 1.

While qubit allocation is NP-hard for arbitrary connectivity graphs, this can be shown easily by a reduction from graph isomorphism [33]. However, real-world devices are not arbitrary, and by imposing some restrictions on the connectivity graph, we can solve the problem in polynomial time [7]. For example, the problem is solvable in polynomial time for path graphs, complete graphs, tree graphs, and product graphs. These results has already been known for similar classical problems like token swapping and routing via matching [3].

Problem 5 (Routing via Matching)**.** *Given a graph and a set of pebble, each of them at each node, a permutation is given and we need to achive the permutation by moving the pebbles. each move consists of swapping two pebbles at adjacent nodes. The problem is to find the minimum number of moves (a.k.a. routing number).*

For arbitrary connectivity graphs, there have been attempts to solve the problem, which can be feasible for small devices [33]. The most common approach is to use a heuristic [39, 18, 10] together with a search algorithm (such as BFS, A^*[41], simulated annealing[40], or others[23]) to find a reasonable solution.

In most cases, the treatment of the initial mapping and subsequent mappings is different [40, 23]. This is because while subsequent mappings can be seen as a routing problem in the permutation space, the initial mapping is a search to find the best starting point.

Another approach is to use partial permutations, which allows for the use of the same algorithm for both initial mapping and subsequent mappings [7, 41].

Definition 3 (Partial Mapping)**.** *A partial mapping, is a partial injective function from the logical qubits to the physical qubits. It means that some of the logical qubits may not be mapped to any physical qubit.*

Current quantum compilers, such as those used in Qiskit[30, 11, 25] and Tket[34], use even simpler heuristics, while considering the errors of the device, which is not the case for most papers.

3.1.2 Generic Gate Set

Most of the existing quantum computers need to decompose CNOTs and they are natively using other families of two-qubit gates, such as [15], but, almost all of the results in the literature are based on the CNOT gate, and it has become a standard for the intermediate representation of quantum circuits, along with the set of one-qubit gates [41, 33, 23, 39, 40, 18, 25, 34]. Although it is suboptimal for hardware, it is a theoretically well-studied two-qubit gate.

In addition to CNOT gates, it is also inevitable to use SWAP gates while allocating and routing the logical qubits through the physical qubits. However, there is another technique that could be used in some scenarios, called bridge gates [34, 18, 32, 33] or remote CNOTs [40, 26], which will be studied in detail in the next section.

3.1.3 Routing and Bridging

The solution to the qubit allocation problem will not necessarily specify the SWAPs that are needed to change the mapping, although some approaches do so [7, 23, 40]. In other cases, we need to use a routing or search algorithm to find the SWAPs that are required to change the mapping [41, 34].

Moreover, bridge gates can be used as an alternative in cases where we need to SWAP back and forth between two qubits. While most papers use only bridge gates for three qubits [34, 18, 32, 33] (one qubit in between), the general case of bridge gates is also studied [40, 26].

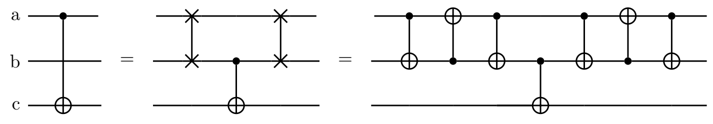


Figure 3.1: Applying a CNOT gate on (a, c) using a SWAP gate

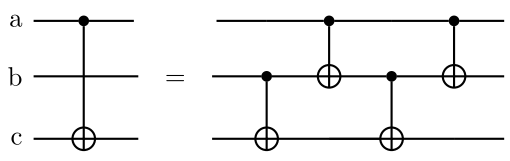


Figure 3.2: Applying a CNOT gate on (a, c) using a bridge gate

Figure 5.2.1 and 5.2.1 show the difference between using a SWAP gate and a bridge gate to perform a CNOT gate for the case of three qubits. The bridge gate is more efficient in terms of the number of CNOT gates, but it is less efficient in terms of depth.

Now, the generalized bridge gate is defined as follows [26]:

Definition 4. *Generalized Bridge Gate A CNOT between qubit* 1 *and* n *can be* *performed using a generalized bridge gate as follows:*

$$\text{Bridge}(1,n) = \prod_{i=1}^{n-1} (\text{CNOT}(i+1,i) \prod_{i=n-2}^{2} \text{CNOT}(i+1,i))^2 \tag{3.1}$$

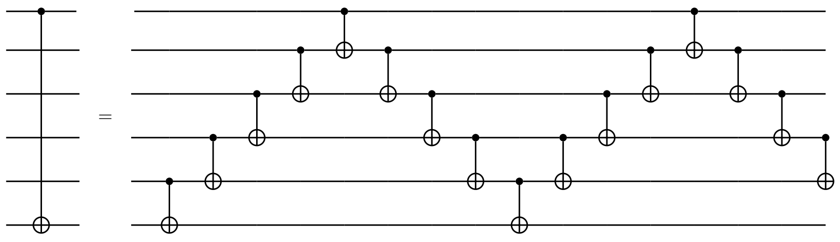


Figure 3.3: The bridge gate for $n = 6$

3.1.4 Circuit Optimization

Circuit optimization is often achieved by applying simplification rules to the circuit [28]. These simplification rules are usually based on the commutation relations of the gates [18].

Yet, this simplification rules are implemented as a pattern matching, therefore the hidden patterns that can be revealed by another simplification rules will not be found.

3.2 Hamiltonian Compilation

The literature on Hamiltonian compilation is currently limited. Even existing compilers have not implemented anything beyond a first-order Suzuki-Trotter decomposition [34, 30]. Also, if we consider the Suzuki-Trotter decomposition as a compiler, there are also a few related works that could be considered as a part of the literature, such as an analysis of its error [8].

Not all efforts in Hamiltonian compilation are based on the Suzuki-Trotter decomposition. Another approach is to use a randomized compiler based on sampling Hamiltonian terms [5]. This method is called QDRIFT protocol and it is briefly described in the theorem below.

Theorem 4 (QDRIFT Compiler)**.** *Given $H = \sum_{j=1}^{k} h_j$ and an oracle that samples the terms of the Hamiltonian by weight $\frac{\|h_i\|_\infty}{\sum_{j=1}^{k} \|h_j\|_\infty}$, by randomly sampling N terms $\eta_1 \ldots \eta_N$, then we can define the approximate evolution as*

$$U(t) = \prod_{i=1}^{N} e^{it \frac{\eta}{N} \sum_{j=1}^{k} \|h_j\|_\infty} \tag{3.2}$$

where it asymptotically outperform first-order Suzuki-Trotter decomposition in number of gates at a certain level of error.

Our baseline compiler is described in [22], which uses the Suzuki-Trotter decomposition and defines routing and scheduling algorithms specifically for Hamiltonian compilation.

The compiler includes two different heuristic search algorithms, one for initial qubit allocation and another for subsequent qubit allocation. The algorithm for subsequent qubit allocation also provides the route (of SWAP gates) for reallocation. This compiler supports all one and two-qubit gates as intermediate gates, which helps simplify the circuit by unifying consecutive two-qubit gates [22].

3.2.1 Scheduling

Even general quantum circuits may have a small degree of freedom in the order of execution. Moreover, the approximate circuit of a Hamiltonian may have a total arbitrariness in the order of execution. This degree of freedom can be set wisely to reduce the complexity of the circuit.

Problem 6 (Scheduling)**.** *Given a quantum circuit that has a DAG of dependencies, the scheduling problem seeks for an optimal order of execution of the gates.*

Several heuristic algorithms have been proposed in the literature for both Hamiltonians [22] and the general case [40, 41].

Chapter 4

Open Problems and Related Questions

We encountered several open problems and related questions while working on Hamiltonian compilation, which are listed below along with brief notes on our thoughts about them.

- **Optimality of Bridge Gates**: It has been shown that the bridge gate for n qubits has a depth and number of CNOT gates of $4n+O(1)$. This is equivalent to $3n+O(1)$ and $6n+O(1)$, respectively, for the naive SWAP gates, indicating that one of them is better in terms of depth and the other outperforms in the number of CNOT gates.

  We have already shown that there is another design for bridge gates with $4n+O(1)$ and $n+O(1)$, and furthermore, it is optimal.

- **Bridging Other Two-Qubit Gates**: The bridge gate is currently only defined for CNOT gates. The question is whether it is possible to define a bridge gate for all two-qubit gates.

  We have shown that there is a SWAP bridge gate for three qubits that requires 8 CNOT gates, rather than the naive 9 CNOT gates. Surprisingly, it could not be generalized, and the naive SWAPs are asymptotically optimal.

- **Defining Problems on Hamiltonian Graph**: We already know that a device graph, due to the restrictions imposed by the physical hardware, is not much like an arbitrary graph and could be represented by simpler structures than graphs (like indices of the lattice, etc.). For the compilation of Hamiltonians, we also have another graph, the Hamiltonian graph, which is a hypergraph with a hyperedge for each Hamiltonian term. If we can define the problems on the Hamiltonian graph rather than the device graph, it may be easier to solve them.

Currently, we are working on Hamiltonian graph coloring as a way to group the gates of Suzuki-Trotter decomposition into layers. We are hopeful that this approach will help solve allocation and scheduling subproblems.

Chapter 5

Discussion and Results

The problem of Hamiltonian compilation could be define rigorously as follows.

Problem 7 (Hamiltonian Compilation)**.** *Given a Hamiltonian H and a device graph G, find a circuit C that approximates the evolution of H on G.*

Generally,

As like as the classical compilation, the quantum compilation problem, where we here narrow it down to the Hamiltonian compilation, is a complex problem that requires a breakdown in order to analyze it properly. In this chapter, we first discuss the problem and its subproblems, then we present our improvements on the baseline results of the literature.

Here we introduce a hardware-aware compiler, starts from a 2-local Hamiltonian, defined in Definition 6, allocates qubits and generates intermediate gates that are studied in Section **??** and finally transpile them to a lower-level gate-set. This compiler aims to reduce error in a reasonable compile-time, Therefore a detailed description of the assumed error model is given in Section 5.3.

In the literature of quantum computing, there is an implementation of CNOT gate for non-adjacent qubits that share a neighbour, which is called bridge gate [32, 18]. This implementation is shown in Figure **??**.

The importance of such an implementation will be clear when we compare it to the baseline implementation, using the SWAP gates. The baseline implementation is shown in Figure 5.

In this paper, we study the possible extenstion the bridge implementation for an arbitrary distance and arbitrary two-qubit gate.

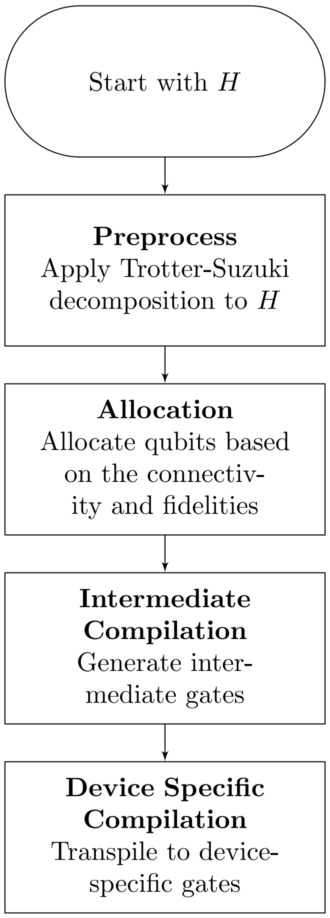

Start with H
Preprocess
Apply Trotter-Suzuki decomposition to H
Allocation
Allocate qubits based on the connectivity and fidelities
Intermediate Compilation
Generate intermediate gates
Device Specific Compilation
Transpile to device-specific gates

Figure 5.1: The overview of the compilation process in 2QANIM

Bridging CNOT gate

Definition 5. *Generalized Bridge Gate For an even n*

$$
\begin{aligned}
\mathrm{Bridge}(1,n) = &\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i+1,i)\mathrm{CNOT}(n-i+1,n-i) \\
&\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i,i+1)\mathrm{CNOT}(n-i,n-i+1) \\
&\mathrm{Bridge}(n/2-1,n/2) \\
&\left(\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i+1,i)\mathrm{CNOT}(n-i+1,n-i)\right)^{\dagger} \\
&\left(\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i,i+1)\mathrm{CNOT}(n-i,n-i+1)\right)^{\dagger}
\end{aligned}
$$

Figure 5.2: The baseline implementation of CNOT gate.

but for an odd n, it is a little bit different.

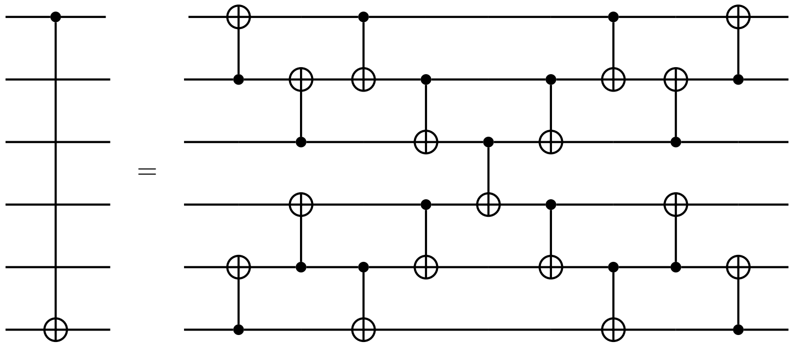


Figure 5.3: The bridge gate for $n = 6$

As far as we know, the family of bridge gates with more than one qubit in the chain is not studied yet, and neither implemented in any quantum compiler. It can be easily shown that for a simple case like the one in the figure **??** that we need to swap and return the qubits, using bridge gate can reduce both depth and number of gates.

Later we prove that this circuit is optimal.

Bridging SWAP gate

Bridging arbitrary gates

For any two-qubit gate U, we can define $R_{\leftarrow}$ and $R_{\rightarrow}$ as the amount of information that can be commuicated back and forth. Note that they may vary for different scenarios of using U, (e.g. different initialization).

We know that

$$\begin{cases} R_{\leftarrow} + R_{\rightarrow} \leq E_U \\ R_{\leftarrow} \leq 1 \\ R_{\rightarrow} \leq 1 \end{cases} \tag{5.1}$$

For a simple case of $n = 1$ intermediate gates, the bridge gate will be a three-qubit gate

$$\mathcal{B} \in \mathcal{H}_a \otimes \mathcal{H}_c^{\otimes n} \otimes \mathcal{H}_b \tag{5.2}$$

And we want the superoperator of the bridge gate, tracing out a and b to be

$$\mathrm{Tr}_{a,b}(\rho) = \mathrm{Tr}_{a,b}(\mathcal{B}\rho\mathcal{B}^{\dagger}) \tag{5.3}$$

Which means that for $\mathcal{B} = \prod U_i V_i$ where

$$U_i = (U_{ia} \otimes U_{ic}) e^{i\theta \hat{n} \cdot \vec{\sigma_{ac}}} (U'_{ia} \otimes U'_{ic}) \tag{5.4}$$

$$V_i = (V_{ic} \otimes V_{ib}) e^{i\theta \hat{n} \cdot \vec{\sigma_{cb}}} (V'_{ic} \otimes V'_{ib}) \tag{5.5}$$

we can write

$$\mathrm{Tr}_{a,b}(\rho) = \mathrm{Tr}_{a,b}(\prod U_i V_i \rho \prod V_i^\dagger U_i^\dagger) \tag{5.6}$$

$$= \mathrm{Tr}_{a,b}(e^{i\theta \hat{n} \cdot \vec{\sigma_{ac}}} (U'_{ic} V_{ic}) e^{i\theta \hat{n} \cdot \vec{\sigma_{cb}}} \rho \text{h.c. }) \tag{5.7}$$

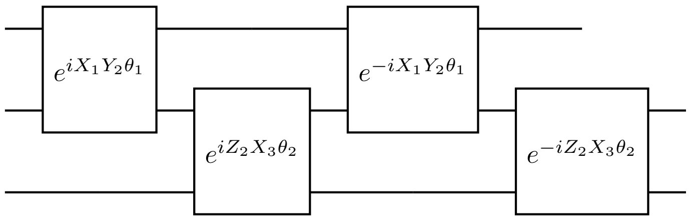
In another words

$$e^{H(\alpha,\beta,\gamma)} = (\cos(\alpha) + iXX\sin(\alpha))(\cos(\beta) + iYY\sin(\beta))(\cos(\gamma) + iZZ\sin(\gamma)) \tag{5.8}$$

$$= \cos(\alpha)\cos(\beta)\cos(\gamma) + iXX\sin(\alpha)\cos(\beta)\cos(\gamma) + iYY\cos(\alpha)\sin(\beta)\cos(\gamma) \tag{5.9}$$

$$+ iZZ\cos(\alpha)\cos(\beta)\sin(\gamma) + ZZ\sin(\alpha)\sin(\beta)\cos(\gamma) + YY\sin(\alpha)\cos(\beta)\sin(\gamma) \tag{5.10}$$

$$+ XX\cos(\alpha)\sin(\beta)\sin(\gamma) + i\sin(\alpha)\sin(\beta)\sin(\gamma) \tag{5.11}$$



We can simplify this cicuit as

$$\text{circuit} = e^{-iZ_2X_3\theta_2}e^{-iX_1Y_2\theta_1}e^{iZ_2X_3\theta_2}e^{iX_1Y_2\theta_1} \tag{5.12}$$

$$= (I\cos\theta_2 - iZ_2X_3\sin\theta_2)(I\cos\theta_1 - iX_1Y_2\sin\theta_1)(I\cos\theta_2 + iZ_2X_3\sin\theta_2)(I\cos\theta_1 + iX_1Y_2\sin\theta_1) \tag{5.13}$$

$$= I\cos^2\theta_1\cos^2\theta_2 - iZ_2X_3\sin\theta_2\cos^2\theta_1\cos\theta_2 - iX_1Y_2\sin\theta_1\cos^2\theta_2\cos\theta_1 + iZ_2X_3\sin\theta_2\cos^2\theta_1\cos\theta_2 + \tag{5.14}$$

$$(-1)X_1(-iX_2)X_3\sin\theta_1\sin\theta_2\cos\theta_1\cos\theta_2 + I\cos^2\theta_1\sin^2\theta_2 + X_1(-iX_2)X_3\sin\theta_1\sin\theta_2\cos\theta_1\cos\theta_2 + X_1 \tag{5.15}$$

$$I\sin^2\theta_1\cos^2\theta_2 + (-1)X_1(-iX_2)X_3\sin\theta_1\sin\theta_2\cos\theta_1\cos\theta_2 \tag{5.16}$$

$$- iZ_2X_3\sin^2\theta_1\sin\theta_2\cos\theta_2 + iX_1Y_2\sin^2\theta_2\sin\theta_1\cos\theta_1 - iZ_2X_3\sin^2\theta_1\sin\theta_2\cos\theta_2 + iX_1Y_2\sin^2\theta_2\sin\theta_1 \tag{5.17}$$

$$- I\sin^2\theta_1\sin^2\theta_2 \tag{5.18}$$

$$= I(\cos^2\theta_1\cos^2\theta_2 + \sin^2\theta_1\cos^2\theta_2 + \cos^2\theta_1\sin^2\theta_2 - \sin^2\theta_1\sin^2\theta_2) + \tag{5.19}$$

$$2iX_1Y_2\sin^2\theta_2\sin\theta_1\cos\theta_1 - 2iZ_2X_3\sin^2\theta_1\sin\theta_2\cos\theta_2 \tag{5.20}$$

$$\tag{5.21}$$

5.1 Basic Concepts

Because of the exponential growth of the number of gates in the compilation of a k-local term [?], setting $k = 2$ is a reasonable choice for NISQ devices. This simplification not only helps us by reducing the number of gates, also enables us to use graph theory to define and analyze the problem, which is not easily possible for higher values of k.

Here we introduce our notation for the Hamiltonian.

Definition 6 (2-local Hamiltonian). *Defining an N-qubit Hilbert space $\mathcal{H} := \otimes_{i=1}^{N}\mathcal{H}_\rangle$, a 2-local Hamiltonian H acting on it will be like*

$$H := \sum_{i=1}^{k} h_i \tag{5.22}$$

where each h_i is a 2-local term, acting nontrivially on two specific qubits.

Definition 7 (Graph representation of a Hamiltonian). *We will define the graph $G_H(V_H, E_H)$ to represent the Hamiltonian H, in a way that for each term h_i in H, we have an edge e_i between the two qubits it acts on.*

The very first step toward the compilation of a 2-local Hamiltonian is to apply the first-order Trotter-Suzuki decomposition [36] to it. This decomposition is a way to approximate the time evolution of a Hamiltonian by a sequence of gates.

Definition 8 (Trotter-Suzuki decomposition)**.** *The first order Trotter-Suzuki decomposition, decomposes a Hamiltonian H, into a set of two-qubit gates g_is which associated with an edge like e_i.*

$$\textit{Target Gates} := \{(g_i, e_i)\} \tag{5.23}$$

The key feature of this decomposition (only the first order) is that the resulting gates can be applied in any order. This arbitrariness is not hold for a general quantum circuit and it may cause an advantage for a problem-specific compiler.

5.2 Intermediate Gate Set

In order to breakdown the process of compilation, which is surely a complex task, we use an intermediate description of the circuit, in terms of the intermediate gate set.

This gate set is presumed to not be directly available on any device, but rather, it is easy to generate them relatively efficient on a quantum device. It is not wondering that we assume that the intermediate circuit must obey the topology of the device.

While the implementation of one-qubit gates is not challenging, it is convenient to have all of the one-qubit gates, in the set, but for two qubit gates, many restrictions must be applied.

The most important two-qubit gate that is often a part of any gate set is the CNOT gate. And because of the topology of the device, the CNOT might not be available on all pairs of qubits. Therefore we often add SWAP gate to make it possible all the way through the circuit, but optimality of the circuit is questionable in this case. Here we introduce a new gate, which is a generalization of the CNOT gate, which is shown to be more efficient than applying a SWAP gate.

5.2.1 Introducing Bridge Gates

For a simple case, that we have three qubits, called a, b, c, and we want to apply a CNOT gate on (a, c), but the connectivity only allows us to apply a CNOT gate on (a, b) and (b, c), the first solution would be to use a SWAP gate, which is shown in figure 5.2.1 (7 gates, depth of 7) While another approach is to use a bridge gate, which is shown in figure 5.2.1 (4 gates, depth of 4).

This example, could be generalized and the bridge gate must be generalized as well. Hereby we define the generalized version of the bridge gate and we show the optimality of the bridge gate in terms of the number of CNOT gates and the depth of the circuit.

Definition 9. *Generalized Bridge Gate For an even n*

$$\begin{aligned}\mathrm{Bridge}(1,n) = &\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i+1,i)\mathrm{CNOT}(n-i+1,n-i)\\ &\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i,i+1)\mathrm{CNOT}(n-i,n-i+1)\\ &\mathrm{Bridge}(n/2-1,n/2)\\ &\left(\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i+1,i)\mathrm{CNOT}(n-i+1,n-i)\right)^{\dagger}\\ &\left(\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i,i+1)\mathrm{CNOT}(n-i,n-i+1)\right)^{\dagger}\end{aligned}$$

but for an odd n, it is a little bit different.

Theorem 5. *Assume that we decomposed a CNOT between the first and the last qubits in a chain, in terms of local circuit. This circuit must carry information from the first bit to the last and reverse. For the forward information flow, the information is an X gate placed on the first qubit, will travel along the circuit, so we need to have a consequential n CNOT gates to carry this information. The same argument is valid for a backward series of gates. Ignoring the first and the last qubit, for each qubit this argument is also valid that, the gate in those series must not be the first neither the last CNOT gate applied on the qubit. Therefore, at least we need to have two series, each has to have a range of CNOT gates, before and after. One could easily manage to find out that the simplest structure would be similar (in terms of complexity) to the proposed bridge gate.*

As far as we know, the family of bridge gates with more than one qubit in the chain is not studied yet, and neither implemented in any quantum compiler. It can be easily shown that for a simple case like the one in the figure **??** that we need to swap and return the qubits, using bridge gate can reduce both depth and number of gates.

5.2.2 Unifying Gates

TODO We can combine two-qubit gates...

5.3 Error Model

Here we introduce a model to estimate errors of a circuit on a device. In this model, we take the average infidelity of a circuit as a measure of the error. In order to calculate the average (in)fidelity, we consider two different source of errors.

- **Decoherence**: When a qubit is left alone for some amount of time, it will decay to a random state. The average fidelity of the qubit can be written as:
$$F_{\text{decoherence}}(t) = \exp\left(-\frac{t}{\tau_{\text{decoherence}}}\right) \tag{5.24}$$
- **Gate Errors**: Any gate, specially the two-qubit gates, are not perfect. We can assign an average fidelity to each gate.

As for any parallel or sequential processes, the average fidelity of the circuit is the product of the average fidelity of each process, we use $-\log \mathcal{F}$, as a positive measure of error with additivity.

Therefore, a device is defined as a tuple of the following:

Definition 10 (Device)**.**

$$Device := (w_0, G_d(V_d, E_d, w_{\mathcal{F}}))$$

where
$$\begin{cases} w_0 = \tau_{\textit{2-qubit gate time}} / \tau_{\textit{decoherence}} \\ w_{\mathcal{F}}(e) = -\log\big(\bar{\mathcal{F}}_{\textit{2-qubit gate on edge e}}\big) \end{cases}$$

5.4 Problem Formulation

5.4.1 NP-Hardness

Anyhow we define the problem, it is inevitable to have a subproblem like below

Problem 8 (Qubit Allocation)**.** *Given a graph that represents a Hamiltonian $G_H(V_H, E_H)$, find an allocation $\phi : V_H \to V_d$ such the maximum number of edges in E_H are mapped to E_d.*

We may use any other more complex criteria, such as minimizing estimated circuit error or minimizing the number of gates in the implemented circuits. But even in the simplest case, the problem is NP-hard, as the graph isomorphism problem can be reduced to it [**?**].

But yet, this argument could be easily misintepreted into the fact that it is impossible to address the exact compilation algorithm in any case. It states that for an abitrary device graph, the problem is NP-hard. If we just add a simple constraint that the device graph has a bounded degree, then the problem is solvable in polynomial time. [**?**]

5.4.2 General Treatment

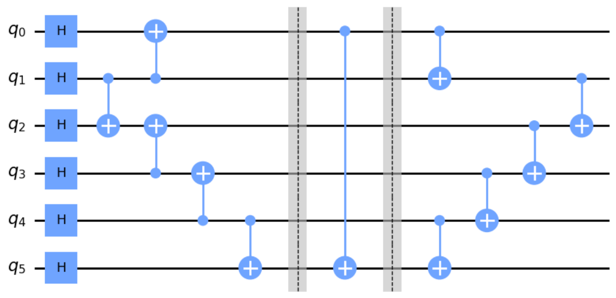
5.5 Algorithms

5.5.1 2QAN

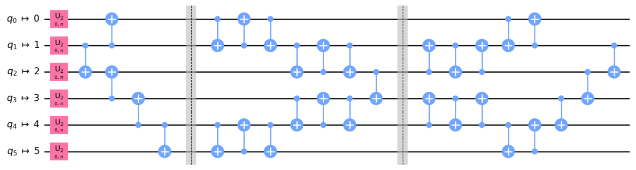
5.5.2 A*

5.5.3 Resource Allocation

a)



b)



c)

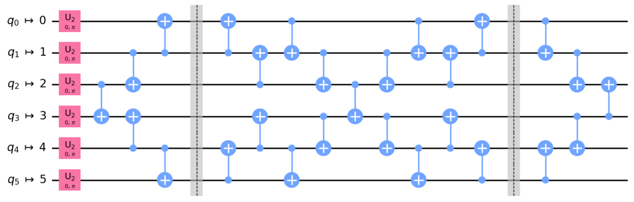


Figure 5.4: a) The original circuit, consisting of some local operations, then a far away CNOT and then some local operations. b) The circuit after transpiling using Qiskit. c) The circuit after transpiling using bridge gate as an intermediate gate.

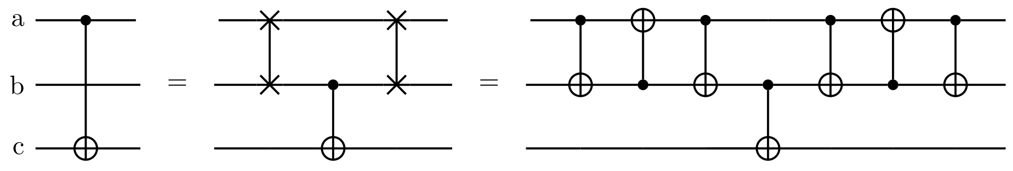


Figure 5.5: Applying a CNOT gate on (a, c) using a SWAP gate

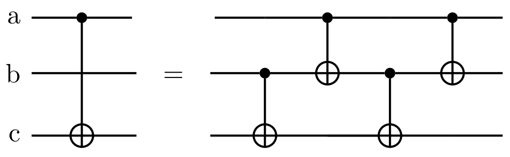


Figure 5.6: Applying a CNOT gate on (a, c) using a bridge gate

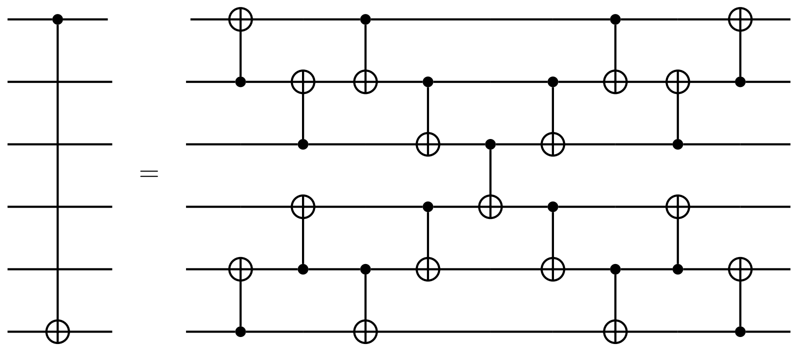
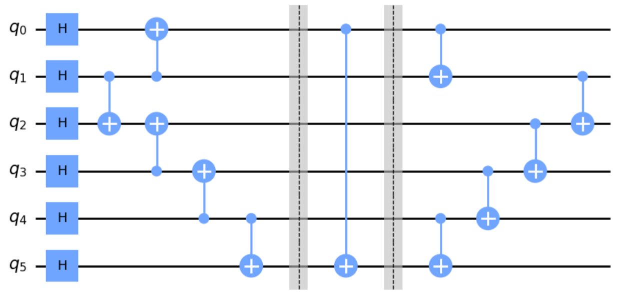
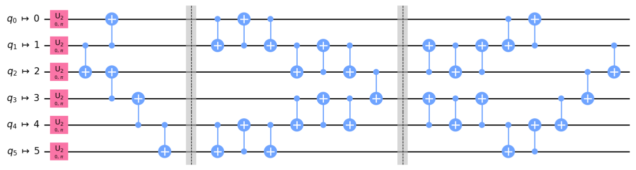


Figure 5.7: The bridge gate for $n = 6$

a)



b)



c)

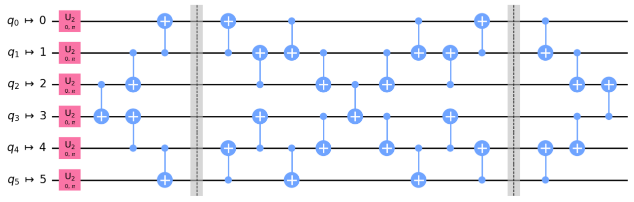


Figure 5.8: a) The original circuit, consisting of some local operations, then a far away CNOT and then some local operations. b) The circuit after transpiling using Qiskit. c) The circuit after transpiling using bridge gate as an intermediate gate.

Chapter 6

Conclusion

In conclusion, quantum compilation is a crucial step towards the realization of useful quantum computations on noisy and intermediate-scale devices. In this essay, we have started with the important tools that are necessary for designing and implmentation of a Hamiltonian compiler. We have also reviewed some of the recent advancements in quantum compilation, in general and Hamiltonian compilation as well. Finally discussed some of the open problems and related questions in the field. We believe that addressing these open problems will help us to improve the performance of the Hamiltonian compilers.

Bibliography

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers - Principles, Techniques and Tools*. English. 1st edition. Reading, Mass: Pearson, Jan. 1986. ISBN: 9780201100884.

[2] Randy Allen and Ken Kennedy. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*. English. 1st edition. San Francisco: Morgan Kaufmann, Oct. 2001. ISBN: 9781558602861.

[3] Indranil Banerjee and Dana Richards. "New Results on Routing via Matchings on Graphs". In: *Fundamentals of Computation Theory*. Springer Berlin Heidelberg, 2017, pp. 69–81. DOI: 10.1007/978-3-662-55751-8_7.

[4] Dominic W. Berry and Barry C. Sanders. "Relation between classical communication capacity and entanglement capability for two-qubit unitary operations". In: *Phys. Rev. A. 68, 032312 (2003)* 68.3 (July 11, 2002), p. 032312. DOI: 10.1103/physreva.68.032312. arXiv: quant-ph/0207065 [quant-ph].

[5] Earl Campbell. "Random Compiler for Fast Hamiltonian Simulation". In: *Physical Review Letters* 123.7 (Aug. 2019), p. 070503. DOI: 10.1103/PhysRevLett.123.070503. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.070503 (visited on 01/22/2023).

[6] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Sept. 2021). ZSCC: 0001030, pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. URL: https://www.nature.com/articles/s42254-021-00348-9 (visited on 03/17/2023).

[7] Andrew M Childs, Eddie Schoute, and Cem M Unsal. "Circuit Transformations for Quantum Architectures". In: (), p. 29.

[8] Andrew M. Childs et al. "A Theory of Trotter Error". In: *Physical Review X* 11.1 (Feb. 1, 2021). ZSCC: 0000002, p. 011020. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.11.011020. arXiv: 1912.08854[cond-mat,physics:physics,physics:quant-ph]. URL: http://arxiv.org/abs/1912.08854 (visited on 03/24/2023).

[9] Andrew M. Childs et al. "Toward the first quantum simulation with quantum speedup". In: *Proceedings of the National Academy of Sciences* 115.38 (Sept. 18, 2018). ZSCC: 0000363, pp. 9456–9461. DOI: 10.1073/pnas.1801723115. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1801723115 (visited on 03/17/2023).

[10] Alexander Cowtan et al. "On the qubit routing problem". en. In: (2019). arXiv:1902.08091 [quant-ph], 32 pages. DOI: 10.4230/LIPIcs.TQC.2019.5. arXiv: 1902.08091 [quant-ph]. URL: http://arxiv.org/abs/1902.08091 (visited on 02/03/2023).

[11] Andrew Cross et al. "OpenQASM 3: A Broader and Deeper Quantum Assembly Language". In: *ACM Transactions on Quantum Computing* 3.3 (Sept. 30, 2022), pp. 1–50. ISSN: 2643-6809, 2643-6817. DOI: 10.1145/3505636. (Visited on 11/11/2022).

[12] Andrew J. Daley et al. "Practical quantum advantage in quantum simulation". In: *Nature* 607.7920 (July 2022). ZSCC: 0000072, pp. 667–676. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04940-6. URL: https://www.nature.com/articles/s41586-022-04940-6 (visited on 03/24/2023).

[13] Edward Farhi and Aram W. Harrow. *Quantum Supremacy through the Quantum Approximate Optimization Algorithm.* arXiv:1602.07674. ZSCC: NoCitationData[s0] type: article. arXiv, Oct. 20, 2019. DOI: 10.48550/arXiv.1602.07674. arXiv: 1602.07674[quant-ph] [quant-ph]. URL: http://arxiv.org/abs/1602.07674 (visited on 03/17/2023).

[14] Richard P. Feynman. "Quantum mechanical computers". In: *Foundations of Physics* 16.6 (June 1, 1986). ZSCC: 0001758, pp. 507–531. ISSN: 1572-9516. DOI: 10.1007/BF01886518. URL: https://doi.org/10.1007/BF01886518 (visited on 03/23/2023).

[15] B. Foxen et al. "Demonstrating a Continuous Set of Two-qubit Gates for Near-term Quantum Algorithms". In: *Phys. Rev. Lett. 125, 120504 (2020)* 125.12 (Jan. 23, 2020), p. 120504. DOI: 10.1103/physrevlett.125.120504. arXiv: 2001.08343 [quant-ph].

[16] Simon Gay and Ian Mackie, eds. *Semantic Techniques in Quantum Computation.* Cambridge University Press, Nov. 2009. DOI: 10.1017/cbo9781139193313.

[17] Robert Hundt. "Quantum Languages, Compilers, and Tools". In: *Quantum Computing for Programmers.* Cambridge University Press, 2022, pp. 292–321. DOI: 10.1017/9781009099974.010.

[18] Toshinari Itoko et al. "Optimization of Quantum Circuit Mapping using Gate Transformation and Commutation". In: (July 2019). DOI: 10.48550/ARXIV.1907.02686. arXiv: 1907.02686 [quant-ph].

[19] S. Jordan. *Quantum Algorithm Zoo.* Mar. 17, 2023. URL: https://quantumalgorithmzoo.org.

[20] B. Kraus and J. I. Cirac. "Optimal Creation of Entanglement Using a Two–Qubit Gate". In: *Physical Review A* 63.6 (May 2001). arXiv:quant-ph/0011050, p. 062309. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.63.062309. URL: http://arxiv.org/abs/quant-ph/0011050 (visited on 02/13/2023).

[21] Matt Langione et al. "Where will quantum computers create value—and when". In: *Boston Consulting Group* (2019), p. 19.

[22] Lingling Lao and Dan E. Browne. *2QAN: A quantum compiler for 2-local qubit Hamiltonian simulation algorithms.* Nov. 7, 2021. arXiv: 2108.02099.

[23] 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.* ASPLOS '19: Architectural Support for Programming Languages and Operating Systems. Providence RI USA: ACM, Apr. 4, 2019, pp. 1001–1014. ISBN: 978-1-4503-6240-5. DOI: 10.1145/3297858.3304023. (Visited on 11/11/2022).

[24] Ashley Montanaro. "Quantum algorithms: an overview". In: *npj Quantum Information* 2.1 (Jan. 2016). DOI: 10.1038/npjqi.2015.23.

[25] Prakash Murali et al. "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.* ACM, Apr. 2019. DOI: 10.1145/3297858.3304075.

[26] Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. "Quantum circuit optimizations for NISQ architectures". In: *Quantum Science and Technology* 5.2 (Mar. 31, 2020). ZSCC: 0000081, p. 025010. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab79b1. arXiv: 1904.01972[quant-ph]. URL: http://arxiv.org/abs/1904.01972 (visited on 03/24/2023).

[27] Michael A. Nielsen et al. "Quantum dynamics as a physical resource". In: *Physical Review A* 67.5 (May 2003), p. 052301. DOI: 10.1103/physreva.67.052301.

[28] Jessica Pointing et al. "Quanto: Optimizing Quantum Circuits with Automatic Generation of Circuit Identities". In: (Nov. 22, 2021). DOI: 10.48550/ARXIV.2111.11387. arXiv: 2111.11387 [quant-ph].

[29] John Preskill. "Quantum computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[30] Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing.* 2023. DOI: 10.5281/zenodo.2573505.

[31] Yi Shen and Lin Chen. "Entangling power of two-qubit unitary operations". In: *Journal of Physics A: Mathematical and Theoretical* 51.39 (Aug. 2018), p. 395303. DOI: 10.1088/1751-8121/aad7cb.

[32] Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov. "Synthesis of Quantum Logic Circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.6 (June 2006). arXiv:quant-ph/0406176, pp. 1000–1010. ISSN: 0278-0070, 1937-4151. DOI: `10.1109/TCAD.2005.855930`. URL: `http://arxiv.org/abs/quant-ph/0406176` (visited on 02/03/2023).

[33] Marcos Yukio Siraichi et al. "Qubit allocation". In: *Proceedings of the 2018 International Symposium on Code Generation and Optimization.* CGO '18: 16th Annual IEEE/ACM International Symposium on Code Generation and Optimization. Vienna Austria: ACM, Feb. 24, 2018, pp. 113–125. ISBN: 978-1-4503-5617-6. DOI: `10.1145/3168822`. (Visited on 11/11/2022).

[34] Seyon Sivarajah et al. "t$—$ket$\rangle$: A Retargetable Compiler for NISQ Devices". In: *Quantum Science and Technology* 6.1 (Jan. 1, 2021). ZSCC: NoCitationData[s0], p. 014003. ISSN: 2058-9565. DOI: `10.1088/2058-9565/ab8e92`. arXiv: `2003.10611[quant-ph]`. URL: `http://arxiv.org/abs/2003.10611` (visited on 03/24/2023).

[35] Masuo Suzuki. "General theory of fractal path integrals with applications to many-body theories and statistical physics". In: *Journal of Mathematical Physics* 32.2 (Feb. 1991), pp. 400–407. DOI: `10.1063/1.529425`.

[36] Hale F Trotter. "On the product of semi-groups of operators". In: *Proceedings of the American Mathematical Society* 10.4 (1959), pp. 545–551.

[37] Farrokh Vatan and Colin Williams. "Optimal Quantum Circuits for General Two-Qubit Gates". In: *Physical Review A* 69.3 (Mar. 22, 2004). ZSCC: 0000332, p. 032315. ISSN: 1050-2947, 1094-1622. DOI: `10.1103/PhysRevA.69.032315`. arXiv: `quant-ph/0308006`. URL: `http://arxiv.org/abs/quant-ph/0308006` (visited on 03/24/2023).

[38] Paolo Zanardi, Christof Zalka, and Lara Faoro. "On the Entangling Power of Quantum Evolutions". In: *Phys.Rev.A62:030301,2000* 62.3 (May 8, 2000), p. 030301. DOI: `10.1103/physreva.62.030301`. arXiv: `quant-ph/0005031 [quant-ph]`.

[39] Chi Zhang et al. "Time-optimal Qubit mapping". In: *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems.* ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. Virtual USA: ACM, Apr. 19, 2021, pp. 360–374. ISBN: 978-1-4503-8317-2. DOI: `10.1145/3445814.3446706`. (Visited on 11/11/2022).

[40] Xiangzhen Zhou, Sanjiang Li, and Yuan Feng. "Quantum Circuit Transformation Based on Simulated Annealing and Heuristic Search". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.12 (Dec. 2020), pp. 4683–4694. ISSN: 0278-0070, 1937-4151. DOI: `10.1109/TCAD.2020.2969647`. URL: `https://ieeexplore.ieee.org/document/8970267/` (visited on 11/11/2022).

[41] Alwin Zulehner, Alexandru Paler, and Robert Wille. “Efficient mapping of quantum circuits to the IBM QX architectures”. In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). Dresden, Germany: IEEE, Mar. 2018, pp. 1135–1138. ISBN: 978-3-9819263-0-9. DOI: `10.23919/DATE.2018.8342181`. URL: `http://ieeexplore.ieee.org/document/8342181/` (visited on 11/11/2022).