Bridging gates over qubits

Seyed Sajad Kahani
22222815
Supervisor: Prof. Dan Browne

August 10, 2023

Contents

Abstract

Chapter 1

Introduction

Quantum computation (QC) 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 [13], 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 [26], 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. The current situation is commonly referred to as the "noisy intermediate scale quantum" (NISQ) era [32], characterized by restricted resources, including a limited number of qubits, constrained qubit connectivity, hardware-specific gate sets, and limited circuit depth due to noise [12].

The restricted qubit resources and excessive noise susceptibility of NISQ devices necessitate optimal compilers to have any hope of useful near-term quantum computation. A huge amount of research has been conducted to tackle different aspects of the compilation problem, including qubit allocation [18, 38, 30, 47, 24], routing [8, 48, 18, 10, 28, 21] and gate synthesis [35, 43, 42, 44, 36, 20]. These aspects are deeply intertwined and one may not distinguish between them, but all of them are in some sense a circuit transformation from a higher-level circuit (with fewer imposed constraints) to a lower-level circuit (with more imposed constraints) [17].

While the knowledge of classical compilation is adopted and the divergent points (no-cloning [**TODO**] and reversibility [**TODO**]) are studied and addressed well, another important distinction has received less attention - the cost of SWAP operations. In classical circuit synthesis, SWAPs simply rearrange wires at negligible cost, compared to two-bit gates. But in the quantum realm, SWAP gates require double entangling [**TODO**] interactions between qubits,

making them the most expensive two-qubit quantum logic gates [**TODO**].

Despite extensive research into minimizing the overall number of SWAP operations [**TODO**, 8], there is little work addressing the inherent cost of each SWAP gate. A few recent works have proposed techniques to reduce the cost of SWAP gates, such as embedding SWAPs within other 2-qubit gates in 2QAN compiler [23], or optimization of SWAP decompositions into CNOT gates [21, 28]. In this work, we aim to address the primary usage of SWAP gates - enabling connectivity between non-adjacent qubits. We analyze the possibility of simplifications for different connectivity cases.

Here we define a problem called bridging that is to find a circuit that applies a two-qubit gate on two non-adjacent qubits. By utilizing the framework of [21] and the extensive literature of network coding [**TODO**] show that in the classical case, the cost of bridging over n bits is $4n$ to $6n$ (upto a $O(1)$ constant). We also present a circuit that achieves the lower bounds. We then attempt to extend the results to the quantum regime, by presenting a circuit to bridge two-qubit gates with Schmidt number 2 over n with optimal number of CNOTs.

This advancement will lead to 33% reduction in the cost of the most expensive two-qubit gate in many situations. To demonstrate the practicality of our results, we implement the algorithms and benchmark with application-oriented dataset of circuits.

The rest of this thesis is organized as follows. Chapter 2 reviews the related works. In Chapter 3, we present the algorithms and prove their correctness and optimality for the classical and quantum cases. We implement the algorithms and benchmark against state-of-the-art techniques in Chapter 4. Finally, Chapter 5 concludes and discusses avenues for future work.

Chapter 2

Background

We should be concerned with two key findings from the literature review on quantum compilation. First, the way that they broke down the problem and the assumption they made to simplify the problem space and introducing some structure to that. On the other hand, is the alogrithms that they involve techniques from QC, graph theory and more.

Here we try to review both of these aspects to draw a big picture of our notion of quantum compilation and also to review the existing techniques related to our approach.

We will also briefly discuss network coding and classical compilation for the purpose of self-containedness.

2.1 Quantum Compilation

The term quantum compilation can be referred any process that transforms a higher-level description of a quantum algorithm into a lower-level description [17]. Given the current situation in the most of works [**TODO**], circuits are the description used and the compilation hence is the process of transforming a general quantum circuit into a circuit that is compatible with a specific hardware. As the problem of finding the most optimal circuit (with respect to a sense of complexity like depth or number of gates) is proven to be NP-hard [**TODO**], the main focus of the research is to break down the problem into smaller problems and to develop techniques to reasonably trade-off between the extension of the process and the quality of the solution. This is pretty much the same as the approach that is taken in classical compilation [**TODO**].

The breakdown structure still is not well-established and there are many different approaches [**<empty citation>**] hence we try to sketch a general picture of the problem and the common patterns in the literature. Afterward we use this general picture as a reference for the rest of this thesis.

To define such picure, we need to the define the circuit transformation as a process that preserves the semantics of the circuit, i.e. the output of the circuit

on any input will be the same before and after the transformation, while it changes the circuit in order to follow some constraints or to optimize the circuit for some goal.

Then, any quantum compilation process can be seen as a series of circuit transformations that each of them is either imposing a constraint or optimizing the circuit (an optimization itself could be seen as a soft constraint [**<empty citation>**]). The main constraints that are imposed on the circuit are coming from the hardware and are listed below:

- **Gate set**: The set of gates physically available in the hardware.
- **Connectivity**: The connectivity between qubits in the hardware topology.
- **Scheduling**: The timing and parallelizability of gate operations.

And the optimizations with respect to different degrees of freedom (like assigning qubits, or choosing between equivalent subcircuits). Commonly the goals are one of these:

- **Complexity**: The number of gates or the depth of the circuit.
- **Soft constraints**: The soft constraints associated with one of the constraints above, to be done before the hard constraint is imposed.

For example, qiskit™, defines the compilation (transpilation) process, as a series of six passes [**TODO**]. That corresponds to our general picture as follows:

1. Init - Unrolling custom gates into one- and two-qubit gates (imposing constraint of gate set with an intermediate gate set).
2. Layout - Maps logical to physical qubits. (Optimization for connectivity constraints)
3. Routing - Inserts swap gates to satisfy connectivity. (Imposing connectivity constraints)
4. Translation - Converts to target gate set. (Imposing gate set constraints)
5. Optimization - Iteratively optimizes until depth limit reached. (Optimization for reducing two-qubit gates)
6. Scheduling - Hardware-aware scheduling. (Imposing scheduling constraints)

While, the most common order is always to impose connectivity, then gateset and then scheduling constraints [**TODO**], there are some exceptions. For example, the work of [45] assumes a different order, and asks all of the constraints to be resloved before the connectivity constraints.

With this background established, we now dive into details on gate set constraints, connectivity constraints, scheduling constraints, and circuit optimizations techniques for reducing complexity.

2.1.1 Gate synthesis

Imposing the gate set constraints which is known as the gate synthesis, is one of the oldest subroutines in the quantum compilation. The problem here is to decompose a general n-qubit gate into a sequence of gates from a universal gate set. It has been proven in the early days of quantum computing that the set of all one-qubit gates (which can be reduced to 2 gates for approximate synthesis [**TODO**]) with CNOT and TOFFOLI is a choice for universal gate set [4]. Afterward, the quest for finding the shortest sequence of gates from this universal gate set for a given gate has been a hot topic in the literature [35, 43].

While there are many other possible choices for universal gate sets [**TODO**], and also most of the existing quantum computers need to decompose CNOTs and they are natively using other families of two-qubit gates, such as [14], still, most 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 [49, 38, 24, 47, 48, 18, 27, 39]. Although it may be suboptimal for hardware, it is a theoretically well-studied two-qubit gate.

Hamiltonian Compilation

[23, 7, 9]

The literature on Hamiltonian compilation is currently limited. Even existing compilers have not implemented anything beyond a first-order Suzuki-Trotter decomposition [39, 33]. 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 [9].

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 [7]. This method is called QDRIFT protocol and it is briefly described in the theorem below.

Theorem 1 (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 $\eta_1 \dots \eta_N$, then we can define the approximate evolution as*

$$U(t) = \prod_{i=1}^{N} e^{it\frac{\eta}{N}\sum_{j=1}^{k}\|h_j\|_\infty} \tag{2.1}$$

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 [23], 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 [23].

2.1.2 Qubit allocation

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

Problem 1 (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 3.

While qubit allocation is NP-hard for arbitrary connectivity graphs, this can be shown easily by a reduction from graph isomorphism [38]. However, real-world devices are not arbitrary, and by imposing some restrictions on the connectivity graph, we can solve the problem in polynomial time [8]. 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 2 (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 [38]. The most common approach is to use a heuristic [47, 18, 10] together with a search algorithm (such as BFS, A^*[49], simulated annealing[48], or others[24]) to find a reasonable solution.

In most cases, the treatment of the initial mapping and subsequent mappings is different [48, 24]. 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 [8, 49].

Definition 1 (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[33, 11, 27] and Tket[39], use even simpler heuristics, while considering the errors of the device, which is not the case for most papers.

2.1.3 Routing

2.1.4 Scheduling

2.1.5 Optimizations

Circuit optimization is often achieved by applying simplification rules to the circuit [31]. 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.

TODO:

- Papers with a focus on inital mapping [38, 47, 30]
- Papers with a focus on routing problem [8, 48, 18, 10]
- Ideas to improve SWAP complexity (embedding from 2QAN and CNOT framework from Kissinger) [23][28, 21]
- ~~Papers with a focus on decomposition of gates and KAK~~ (maybe) [42, 44]
- Papers with a focus on entangling power (maybe) [29, 6, 5, 25]
- A bit from Network coding [16]

2.2 Previous Version (From Literature Review)

2.2.1 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 [15].

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 3 (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 4 (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 5 (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]*

2.2.2 Quantum Computation

2.2.3 Suzuki-Trotter Decomposition

The Suzuki-Trotter decomposition [41, 40] 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 2 (Lie Trotter Formula)**.** *For any $H = A + B$,*

$$e^{iH} = \lim_{n\to\infty} (e^{iA/n} e^{iB/n})^n. \tag{2.2}$$

Generally, the decomposition is stated as follows.

Theorem 3 (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*

$$\tilde{U}_1(t) = e^{ih_1 t} \dots e^{ih_k t}, \tag{2.3}$$

$$\tilde{U}_2(t) = e^{ih_1 t/2} \dots e^{ih_k t/2} e^{ih_k t/2} \dots e^{ih_1 t/2}, \tag{2.4}$$

$$\tilde{U}_{2p}(t) = \tilde{U}_{2p-2}(u_p t)^2 \tilde{U}_{2p-2}((1-4u_p)t) \tilde{U}_{2p-2}(u_p t)^2, \tag{2.5}$$

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

Then,

$$\left\| e^{iHt} - \tilde{U}_{2p}(t) \right\| \in O(t^{2p+1}). \tag{2.6}$$

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

2.3 Special topics on two-qubit gates

Another important fact about the gate synthesis of two-qubit gate, which is fruitful for many theoretical purposes is KAK (aka Khaneja-Glaser [20] or Kraus-Cirac [22]) decomposition.??

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 [39, 18, 35, 38] or remote CNOTs [48, 28], which will be studied in detail in the next section.

Kraus-Cirac Decomposition

The Kraus-Cirac decomposition [22] 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 4 (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*

$$U = V_1 \otimes V_2 e^{\alpha X \otimes X + \beta Y \otimes Y + \gamma Z \otimes Z} V_3 \otimes V_4. \tag{2.7}$$

This theorem, together with the basic properties of entanglement power, will also lead to many more results on communication using two-qubit gates [6] and also the universality and optimality of three CNOTs for two-qubit gates [43].

2.3.1 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 [34], or the average amount of them (with respect to a Haar distribution) [46], 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) [29].

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.

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

2.3.2 General Quantum Compilation

Qubit Allocation

Generic Gate Set

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 [8, 24, 48]. In other cases, we need to use a routing or search algorithm to find the SWAPs that are required to change the mapping [49, 39].

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 [39, 18, 35, 38] (one qubit in between), the general case of bridge gates is also studied [48, 28].

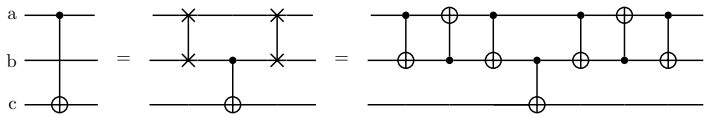

Figure 2.1: Applying a CNOT gate on (a, c) using a SWAP gate

Figure 2.3.2 and 2.3.2 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 [28]:

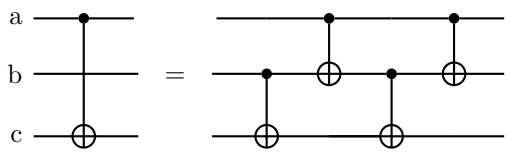

Figure 2.2: Applying a CNOT gate on (a, c) using a bridge gate

Definition 2. *Generalized Bridge Gate A CNOT between qubit* 1 *and* n *can be performed using a generalized bridge gate as follows:*

$$\text{Bridge}(1,n) = \prod_{i=1}^{n-1}(\text{CNOT}(i+1,i) \prod_{i=n-2}^{2} \text{CNOT}(i+1,i))^2 \tag{2.8}$$

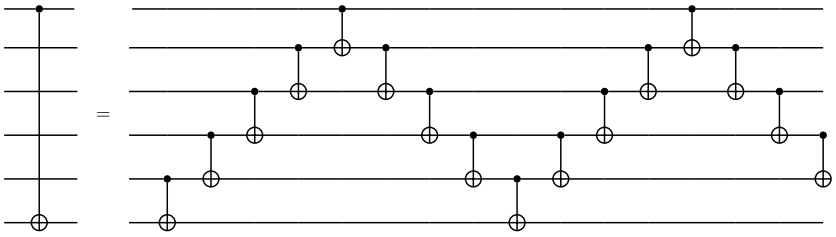

Figure 2.3: The bridge gate for $n = 6$

Circuit Optimization

2.3.3 Hamiltonian Compilation

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 [23] and the general case [48, 49].

Chapter 3

Discussion

While remote gates and ...

It is a well-known fact that a SWAP gate could not be decomposed to less than three CNOTs. For a more complex circuit involving a SWAP or more, this does not necessarily hold in general, especially in presence of connectivity contraints. Here we provide two examples (3.1 and 3.2) with linear connectivity constraints of SWAPs that are used to apply a two-qubit gate on non-adjacent qubits. They show that decomposing SWAPs to three CNOTs will not necessarily lead to an optimal circuit, even in simple cases.

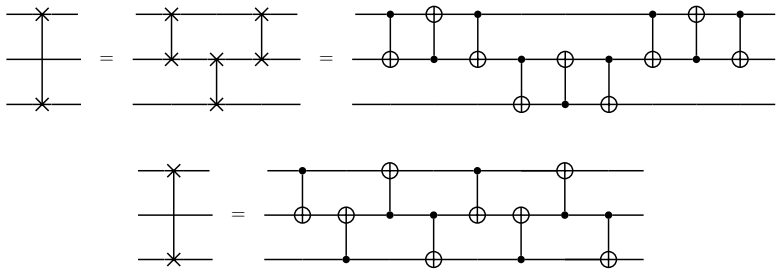

Figure 3.1: Simplifying three SWAP gates

In order to generalize this idea, we define the following problem

Problem 7 (Bridging two-qubit gates)**.** *The solution to the problem of Bridging two-qubit gate U over n qubits is a circuit that can be applied on $n+2$ qubits such that the n qubits in the middle are not affected and the two qubits on the ends are mapped with respect to U. The circuit should normally obey a connectivity and a gate set constraints.*

Note that this definition is valid for classical case as well. In this case, the gate U is a classical reversible gate applying on bits instead of qubits.

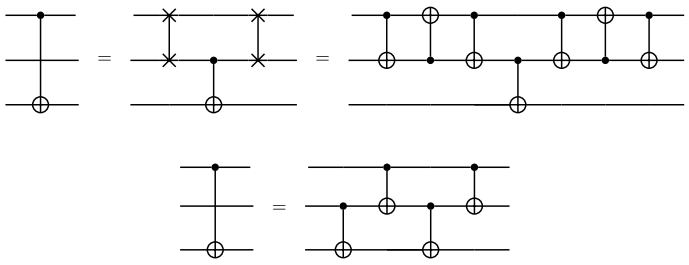

Figure 3.2: Simplifying a CNOT bridge

3.1 Classical Bridging

Quantum logical circuits or classical reversible circuits are a class of circuits that are in common between classical and quantum computation. This class has been defined carefuly [37] and it has been proven that any classical reversible circuit can be synthesized using X (classically known as NOT), CNOT and TOFFOLI gates.

Two-bit classical gates are limited to identity, CNOT (in both directions) and SWAP upto local isomorphism (NOT gates).

We will show that CNOT could be bridged over n qubits using $4n + O(1)$ CNOTs within the depth of $2n + O(1)$. Then, we can prove that not only this is result is optimal, but also SWAP has an optimal bound $3n + O(1)$ CNOTs and $3n + O(1)$ depth that is already fulfilled by a simple decomposition.

Theorem 5. *Bridging CNOT(a, b) over n bits needs at least* $4n$ *CNOTs, where* $n + 1$ *must be the minimum distance between the two bits (a, b).*

Proof. We can layer the graph of bits in a way that the first layer is a, the last layer is b and there are n layers that we name any bit in layer i as c_i. We also define right as the direction towards b and left as th opposite direction.

By considering a $\mathbb{F}_2$ vector space with hamming inner product $\langle \cdot, \cdot \rangle$ over the bits, while each bit at each time has a unique vector, we can see that each CNOT as a linear operation adding a vector to another.

If we name the set of every vector that is associated with a CNOT from layer i to layer $i + 1$ with $F_{i \to i+1}$ and simliarly $F_{i \leftarrow i+1}$, from the fact that the vector a is going to be added to b at the end, we know that the vector a could be extracted from $F_{i \to i+1}$, which means that $a \in \text{span}(F_{i \to i+1})$. On the other hand, in order to layer i be unchanged after the process $\sum_{F_{i \leftarrow i+1}} f + \sum_{F_{i-1 \to i}} f = 0$. Mathematically, we can conclude from these two fact that one of these two cases hold for each layer i:

Case 1 ($P(i)$): $F_{i \to i+1} = D \cup D'$ where $D \cap D' = \emptyset$ and $a \in \text{span}(D)$ and $a \in \text{span}(D')$.

Case 2 ($Q(i)$): exists f that $\langle f, a \rangle = 1$ and $f \in F_{i+1 \leftarrow i+2}$.

We separately prove that for the first case

$$|F_{i\to i+1}| + |F_{i-1\leftarrow i}| \geq 4 \tag{3.1}$$

and for the second case

$$|F_{i+1\to i+2}| + |F_{i+1\leftarrow i+2}| \geq 4 \tag{3.2}$$

Moreover we prove that $P(k+1)$ and $Q(k)$ cannot happen at the same time. This means that if $Q(k)$ holds, $Q(j)$ holds for all $j \geq k$.

Then assuming $Q(k)$ and $P(k-1)$, by summing up the bounds

$$\sum_{i=0}^{k-1} |F_{i\to i+1}| + |F_{i-1\leftarrow i}| + \sum_{i=k}^{n-2} |F_{i+1\to i+2}| + |F_{i+1\leftarrow i+2}| \geq 4n - O(1) \tag{3.3}$$

This is due to the fact that these two summations has $O(1)$ terms in common. This means that the total number of CNOTs is at least $4n + O(1)$. □

Lemma 1 (Bound for $P(i)$)**.** *For the first case used in the proof of Theorem 5, we have*

$$|F_{i\to i+1}| + |F_{i-1\leftarrow i}| \geq 4 \tag{3.4}$$

Proof. If we call any arbitrary bit in ith layer by c_i, we know if $a \in F_{i\to i+1}$,then it means that there was CNOTs, involving in removing c_i from its site and returning it back. By a simple argument of capacity (in bits) the data could only be stored in the left-hand-side of i meaning that vectors with c_i element appeared in each of $F_{i-1\leftarrow i}$ $F_{i-1\to i}$ twice.

Now, we already assumed that $a \in \text{span}(D)$ and $a \in \text{span}(D')$.

Case 1a: If $a \in D$ or $a \in D'$, then $a \in F_{i\to i+1}$. From the argument above we know that the vectors containing c_i appear twice in $F_{i-1\leftarrow i}$. One could easily conclude that $|F_{i\to i+1}| + |F_{i-1\leftarrow i}| \geq 4$.

Case 1b: If $a \notin D$ and $a \notin D'$, then both D and D' need to have two vectors to have a in their span, meaning $|F_{i\to i+1}| \geq 4$. □

Lemma 2 (Bound for $Q(i)$)**.** *For the second case used in the proof of Theorem 5, we have*

$$|F_{i+1\to i+2}| + |F_{i+1\leftarrow i+2}| \geq 4 \tag{3.5}$$

Proof. We know that $f \in F_{i+1\to i+2}$ that $\langle f, a\rangle = 1$, in addition to that, $a \in \text{span}(F_{i\to i+1})$ and independently $a \in \text{span}(F_{i+1\leftarrow i+2})$.

If $\langle f, c_{i+2}\rangle = 0$ it means that c_{i+2} must appear twice in each $F_{i+1\to i+2}$ and $F_{i+1\leftarrow i+2}$, so $F_{i+1\to i+2} + F_{i+1\leftarrow i+2} \geq 4$.

Otherwise if $\langle f, c_{i+2}\rangle = 1$, then c_{i+2} appears once in $F_{i+1\leftarrow i+2}$. Noting the order of CNOTs, it must be before CNOTs associated with a, therefore all of the terms in $F_{i+1\to i+2}$ with a must have c_{i+2} component as well. This makes impossible to $a \in F_{i+1\to i+2}$. So $a \in \text{span}(F_{i+1\to i+2})$ results in a appears twice, so $|F_{i+1\to i+2}| \geq 2$. Putting all together, $|F_{i+1\to i+2}| + |F_{i+1\leftarrow i+2}| \geq 4$. □

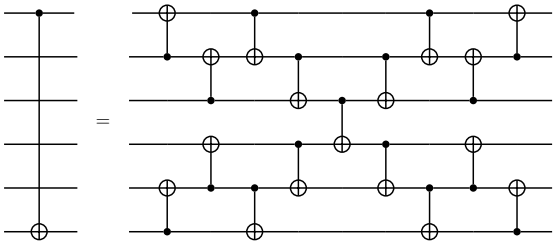

Figure 3.3: Optimal bridging CNOT

It could be easily shown that the circuit that is visualized in Figure 3.3 is an optimal solution for bridging CNOT.

TODO: Show that two TOFFOLIs (with a NOT in between) are equal to a bridged CNOT over one bit.

TODO: Prove the optimality in presence of TOFFOLI in gate set.

Theorem 6. *Bridging SWAP(a, b) over n bits needs at least* $6n$ *CNOTs, where* $n+1$ *must be the minimum distance between the two bits (a, b).*

Proof. TODO □

3.2 Quantum Bridging

We use a operator propagation framework.

- A local opreation does not move the operator.
- A CR_x extends operator to two-qubit but doesn't move it (lemmas)

Lemma 3. *In a two-qubit Hilbert space* $\mathcal{H}_A \otimes \mathcal{H}_B$ *if* $V^{(a)}U^{(a,b)} = U^{(a,b)}V'^{(b)}$ *for any* V*, then* U *has two* CR_x *gates.*

Lemma 4. *In a two-qubit Hilbert space* $\mathcal{H}_A \otimes \mathcal{H}_B$ *if* $V^{(a)}U^{(a,b)} = U^{(a,b)}V'^{(b)}$ *and* $V^{(b)}U^{(a,b)} = U^{(a,b)}V'^{(a)}$ *each for any* V*, then* U *has three* CR_x *gates.*

TODO: some descriptions and text

Lemma 5. *Any two-qubit gate with Schmidt number* 2 *could be written as* $L_1 \otimes L_2 CR_x(\theta) L_3 \otimes L_4$ *where* L_i *are local unitaries and* $CR_x(\theta)$ *is a controlled* R_x *gate.*

Proof. Any two-qubit gate with Schmidt number 2 could be written as $L_1 \otimes L_2 e^{i\alpha ZZ} L_3 \otimes L_4$. Ignoring the local operations, by applying $R_z(-\theta)$ on the first and the second qubit, we will have

$$R_z(-\theta) \otimes R_z(-\theta) e^{i\alpha ZZ} = CR_z(3\theta) \tag{3.6}$$

that could be converted to $CR_x(\theta)$ by applying H on the second qubit. □

Theorem 7 (Optimal bridging For CR_x)**.** *Bridging CR_x over n qubits needs at least $4n$ CNOTs.*

Proof. TODO □

The circuit below, shows how to bridge any two-qubit gate with Schmidt number 2 over n qubits using $4n$ CNOTs and a CR_x. TODO: the shape (similar to CNOT)

3.2.1 Previous Version

Theorem 8 (X-shaped bridge)**.** *Any X-shaped bridge, needs at least $4n$ CNOTs.*

Proof: As we have already defined the bridge gate for CNOT, we can extend it to any two-qubit gate. Therefore a bridge gate is a circuit consisting of two-qubit gates that acts like $U \in \mathcal{H}^{\otimes 2}$ on the first and last qubits, and the identity on the rest of the qubits.

Note that from a simple argument of light cone, we can conclude that the bridge gate must have two chains of $(1,2),\ldots,(n-1,n)$ and $(n-1,n),\ldots,(1,2)$ gates.

This two chain may intercept (by the definition below) at any qubit between 1 and n, making one of these shapes

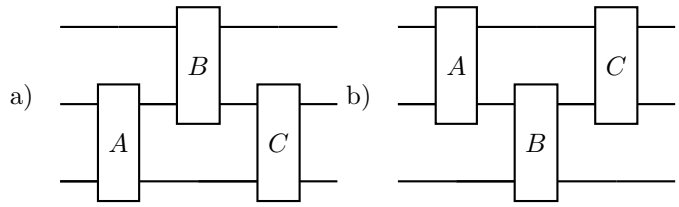

Figure 3.4: Possible intersections of two chains of the bridge gate

We stick to the case b, and we know that they are equivalent upto changing numbers.

We know that any two-qubit gate (such as B) could be decomposed into

$$B = (L_2 \otimes L_3)(\alpha^{(1)} I_2 I_3 + \alpha^{(2)} X_2 X_3 + \alpha^{(3)} Y_2 Y_3 + \alpha^{(4)} Z_2 Z_3)(L'_2 \otimes L'_3) \quad (3.7)$$

where L_i and L'_i are local unitaries on the ith qubit.

Then, the whole circuit ABC will be equal to

$$ABC = \alpha^{(1)} A L_2 (I_1 I_2) L'_2 C \otimes L_3 I_3 L'_3 + \alpha^{(2)} A L_2 (I_1 X_2) L'_2 C \otimes L_3 X_3 L'_3 + \ldots \quad (3.8)$$

And our final goal is to make a bridge that acts like this for all U_2s

$$U_2 ABC = ABC U_2 \quad (3.9)$$

By applying $P_3^{(i)} L_3^\dagger$ and ${L'_3}^\dagger$ from left and right and then applying Tr_3

$$U_2 \alpha^{(i)} A L_2 (I_1 \otimes P^{(i)}) L'_2 C = \alpha^{(i)} A L_2 (I_1 \otimes P^{(i)}) L'_2 C U_2 \tag{3.10}$$

Which directly implies

$$A L_2 (I_1 \otimes P^{(i)}) L'_2 C = V^{(i)} \otimes I \tag{3.11}$$

Note that for all $j, k \in \{1, 2, 3\}$

$$V^{(j)} V^{(0)\dagger} V_{(k)} V^{(0)\dagger} = A L_2 (I_1 \otimes P^{(i)} P^{(j)}) L_2^\dagger A^\dagger = \sum_k i \epsilon_{jkl} V^{(l)} V^{(0)\dagger} \tag{3.12}$$

Here we use a lemma,

Lemma 6 (Rotated Paulis)**.** *If $V^{(j)}$s are unitaries that*

$$V^{(j)} {V^{(0)}}^\dagger V^{(k)} {V^{(0)}}^\dagger = \delta_{jk} I + i \epsilon_{jkl} V^{(l)} {V^{(0)}}^\dagger \tag{3.13}$$

then

$$V^{(j)} = U P^{(j)} U^\dagger {V^{(0)}}^\dagger \tag{3.14}$$

that U is a unitary operation and $P^{(j)}$s are Pauli matrices.

Proof. If we define $Q^{(j)} = V^{(j)} {V^{(0)}}^\dagger$, then we have shown that $Q^{(j)}$s are hermitian unitaries that $Q^{(0)} = I$ and $Q^{(j)} Q^{(k)} = \delta_{jk} I + i \epsilon_{jkl} Q^{(k)}$, then $Q^{(j)} = U P^{(j)} U^\dagger$ for some U. Therefore, $V^{(j)} = U P^{(j)} U^\dagger {V^{(0)}}^\dagger$ □

So far we have shown that

$$A L_2 (I_1 \otimes P^{(i)}) L'_2 C = U P^{(i)} U^\dagger {V^{(0)}}^\dagger \otimes I \tag{3.15}$$

Then, by defining

$$\begin{cases} S_{12} = U^\dagger A L_2 \\ S'_{12} = L'_2 C V^{(0)} U \end{cases} \tag{3.16}$$

we can say that

$$S_{12} (I_1 \otimes P^{(i)}) S'_{12} = P^{(i)} \otimes I \tag{3.17}$$

Now, if we define swap-pair set

Definition 3 (Swap-Pair Set)**.** *swap-pair set for any unitary operation U is defined as*

$$\mathrm{SPS}(U) = \{(X, Y) | X (I \otimes U) Y = U \otimes I\} \tag{3.18}$$

Lemma 7 (Swap-Pair Set for Z)**.** *Any pair that belongs to $\mathrm{SPS}(Z) \cap \mathrm{SPS}(I)$ can be written as* (SWAP W CV, *h.c.*) *for some single-qubit gate W and a controlled-V gate.*

Proof. Note that it is easy to show that $\mathrm{SPS}(I) = \{(S, S^\dagger)|\forall S\}$ and Also, we can show that if $(S, S^\dagger) \in \mathrm{SPS}(Z)$

$$\begin{aligned} & S(I \otimes Z) = (Z \otimes I)S \\ \rightarrow & \begin{cases} S\left|00\right\rangle = (Z \otimes I)S\left|00\right\rangle \\ -S\left|01\right\rangle = (Z \otimes I)S\left|01\right\rangle \\ S\left|10\right\rangle = (Z \otimes I)S\left|10\right\rangle \\ -S\left|11\right\rangle = (Z \otimes I)S\left|11\right\rangle \end{cases} \\ \rightarrow & \begin{cases} S\left|00\right\rangle = \left|0\right\rangle\left|\psi_{00}\right\rangle \\ S\left|01\right\rangle = \left|1\right\rangle\left|\psi_{01}\right\rangle \\ S\left|10\right\rangle = \left|0\right\rangle\left|\psi_{10}\right\rangle \\ S\left|11\right\rangle = \left|1\right\rangle\left|\psi_{11}\right\rangle \end{cases} \end{aligned} \tag{3.19}$$

Noting that $\langle\psi_{00}|\psi_{10}\rangle$ and $\langle\psi_{01}|\psi_{11}\rangle$ are zero, there are these two maps then

$$\begin{cases} W\left|\psi_{00}\right\rangle = \left|0\right\rangle \\ W\left|\psi_{10}\right\rangle = \left|1\right\rangle \\ WV\left|\psi_{01}\right\rangle = \left|0\right\rangle \\ WV\left|\psi_{11}\right\rangle = \left|1\right\rangle \end{cases} \tag{3.20}$$

Finally,

$$S = \mathrm{SWAP}_{1,2}\ W\ \mathrm{CV}_{1\rightarrow 2} \tag{3.21}$$

□

Now, we need to use the lemma to show that S_{12} and S'_{12} are made of 2 or 3 CNOTs.

$$(S, S') \in \mathrm{SPS}(P_i)\forall i \tag{3.22}$$

which means that

$$\begin{cases} U^\dagger A L_2 = \mathrm{SWAP}_{1,2}\ W\ \mathrm{CV}_{1\rightarrow 2} \\ L'_2 C V^{(0)} U = {\mathrm{CV}^\dagger}_{1\rightarrow 2}\ W^\dagger\ \mathrm{SWAP}_{1,2} \end{cases} \tag{3.23}$$

which clearly shows that A and C are made of 2 or 3 CNOT and have rank 4.

By induction, we can expand this argument to all of the gates in the circuit, showing th lower bound of $4n$ CNOTs any bridging.

Chapter 4

Implementation

TODO:

- Qiskit: implementation in the routing stage `https://qiskit.org/documentation/apidoc/transpiler.html#routing-stage`
- tket: just test if it can do any of these simplifications
- PyZX (can do some of these simplifications)

Chapter 5

Conclusion

TODO

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] A. Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* 52.5 (Nov. 1, 1995). ZSCC: 0005107, pp. 3457–3467. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.52.3457. arXiv: quant-ph/9503016. URL: http://arxiv.org/abs/quant-ph/9503016 (visited on 08/10/2023).

[5] C. H. Bennett et al. "On the capacities of bipartite Hamiltonians and unitary gates". In: *IEEE Trans. Inf. Theory, Vol. 49, No. 8, (2003) p.1895-1911* 49.8 (May 10, 2002), pp. 1895–1911. DOI: 10.1109/tit.2003.814935. arXiv: quant-ph/0205057 [quant-ph].

[6] 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].

[7] 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).

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

[9] 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).

[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 W. Cross et al. "Validating quantum computers using randomized model circuits". In: *Phys. Rev. A* 100 (3 Sept. 2019), p. 032328. DOI: 10.1103/PhysRevA.100.032328. URL: https://link.aps.org/doi/10.1103/PhysRevA.100.032328.

[13] 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).

[14] 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].

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

[16] Tracey Ho and Desmond Lun. *Network coding: an introduction*. Cambridge University Press, 2008.

[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] Navin Khaneja and Steffen J. Glaser. "Cartan decomposition of SU(2n) and control of spin systems". In: *Chemical Physics* 267.1-3 (June 2001), pp. 11–23. DOI: 10.1016/s0301-0104(01)00318-4.

[21] Aleks Kissinger and Arianne Meijer-van de Griend. *CNOT circuit extraction for topologically-constrained quantum memories*. arXiv:1904.00633. ZSCC: 0000062 type: article. arXiv, May 26, 2019. arXiv: 1904.00633[quant-ph]. URL: http://arxiv.org/abs/1904.00633 (visited on 07/19/2023).

[22] 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).

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

[24] 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).

[25] Tomasz Linowski, Grzegorz Rajchel-Mieldzioć, and Karol Życzkowski. "Entangling power of multipartite unitary gates". In: *Journal of Physics A: Mathematical and Theoretical* 53.12 (Mar. 2020), p. 125303. DOI: 10.1088/1751-8121/ab749a.

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

[27] 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.

[28] 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).

[29] 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.

[30] Alexandru Paler. *On the Influence of Initial Qubit Placement During NISQ Circuit Compilation*. Jan. 30, 2019. arXiv: 1811.08985.

[31] 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].

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

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

[34] 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.

[35] 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).

[36] Vivek V. Shende, Igor L. Markov, and Stephen S. Bullock. *Minimal Universal Two-qubit Quantum Circuits*. ZSCC: NoCitationData[s0]. Mar. 12, 2004. DOI: 10.1103/PhysRevA.69.062321. arXiv: quant-ph/0308033. URL: http://arxiv.org/abs/quant-ph/0308033 (visited on 07/25/2023).

[37] Vivek V. Shende et al. *Reversible Logic Circuit Synthesis*. arXiv:quant-ph/0207001. ZSCC: NoCitationData[s0] type: article. arXiv, Feb. 27, 2003. DOI: 10.48550/arXiv.quant-ph/0207001. arXiv: quant-ph/0207001. URL: http://arxiv.org/abs/quant-ph/0207001 (visited on 07/26/2023).

[38] 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).

[39] 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).

[40] 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.

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

[42] Robert R. Tucci. *An Introduction to Cartan's KAK Decomposition for QC Programmers*. arXiv:quant-ph/0507171. ZSCC: 0000059 type: article. arXiv, July 18, 2005. DOI: 10.48550/arXiv.quant-ph/0507171. arXiv: quant-ph/0507171. URL: http://arxiv.org/abs/quant-ph/0507171 (visited on 07/19/2023).

[43] 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).

[44] Farrokh Vatan and Colin P. Williams. *Realization of a General Three-Qubit Quantum Gate*. arXiv:quant-ph/0401178. ZSCC: 0000035 type: article. arXiv, Jan. 30, 2004. DOI: 10.48550/arXiv.quant-ph/0401178. arXiv: quant-ph/0401178. URL: http://arxiv.org/abs/quant-ph/0401178 (visited on 07/19/2023).

[45] Robert Wille, Stefan Hillmich, and Lukas Burgholzer. "Efficient and Correct Compilation of Quantum Circuits". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020 IEEE International Symposium on Circuits and Systems (ISCAS). ZSCC: 0000010 ISSN: 2158-1525. Oct. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180791.

[46] 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].

[47] 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).

[48] 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).

[49] 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).