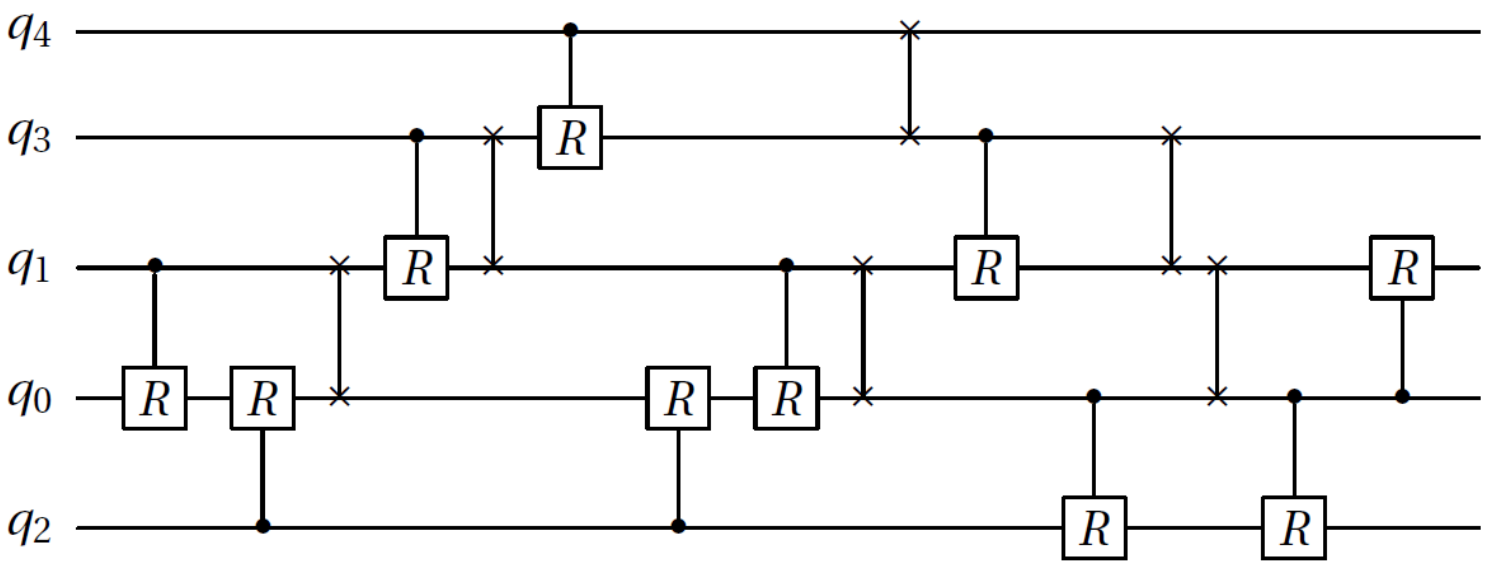
Nearest Neighbor Compliance in Quantum Circuit Design

J. Mulderij

Master Thesis
Applied Mathematics



TU Delft

TNO

Nearest Neighbor Compliance

in Quantum Circuit Design

by

J. Mulderij

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday November 8, 2019 at 02:00 PM.

Student number:	4229754	
Project duration:	February 1, 2019 – November 8, 2019	
Thesis committee:	Prof. dr. ir. K. I. Aardal,	TU Delft, supervisor
	Dr. C. G. Almudever,	TU Delft
	Dr. F. Phillipson,	TNO

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Over the course of the last decade, an interest has emerged in nearest neighbor constraints for quantum circuit design. The challenge herein is to both minimize the running time of a circuit and to modify it such that quantum gates only act on adjacent qubits while leaving the desired computational operation intact. These modifications involve the insertion of SWAP gates, which introduces computational overhead. Several exact and heuristic methods have been developed to determine the minimal number of required SWAP gates. Until now, the model sizes of exact methods scale superexponentionally in the number of qubits of the quantum circuit. The main research goal of this project was to develop an integer linear program that instead scales polynomially both in the number of qubits and the number of quantum gates in the quantum circuit.

First, qubits located on a linear array were considered. The resulting integer linear program proposed requires $\mathcal{O}(n^2 m)$ variables and constraints, where n denotes the number of qubits and m denotes the number of quantum gates of the quantum circuit under consideration. Second, qubits located on a two- or three-dimensional grid were considered. The programs both require $\mathcal{O}(n^4 m)$ variables and $\mathcal{O}(n^3 m)$ constraints.

Subsequently we prove that, for a fixed quantum circuit, the optimal objective value of the one-dimensional problem is an upper bound for the optimal objective value of the corresponding two- and three-dimensional problems. Using CPLEX's Branch & Bound method, 131 benchmark instances of various circuit sizes were evaluated. The largest of which contained either eighteen qubits and sixteen quantum gates, or five qubits and 112 quantum gates.

Furthermore, we identify problems on the interface of nearest neighbor compliant quantum circuit design and distributed quantum computing. Integer linear program formulations are provided for these additional problems.

Preface

Writing three papers during the week and some of the weekends, looking for interesting PhD positions, having progress meetings with my supervisors, listening to quantum related presentations during the Thursday lunch breaks, drinking coffee with the employees and other interns of TNO, driving tanks around and testing new weapons in the lunch breaks at the Oude Waalsdorperweg, walking around the Malieveld during lunch, walking between the tractors during the farmer's protest, playing soccer with TNO during the preparation of a presentation for a graduation drink whilst thinking of a speech for the twenty-one-dinner the next day, breaking out of an escape room with the interns, going to a comedy club in The Hague, going to the TNO beach party, playing beach volleyball, collecting a confused roommate from the hospital, saying goodbye to old roommates and welcoming new ones, catching up with friends in Amersfoort, seeing my brother obtaining his driver's license, brewing up a plan to go parachute jumping together, welcoming my mom and dad in my messy room, teaming up to tidy up my messy room, throwing out old clothes and going shopping with my mom for new ones in Utrecht, going swimming in Ypenburg, sailing over the lakes of Friesland, going paalkamperen in the Dutch forests, hiking on the beaches and through the dunes of Zeeland, going to a Thailand themed house party with backpacks on, celebrating Kingsday in the rain in Breda, playing at a volleyball tournament in even worse rain, seeing Keane live in the cold on the beach at Scheveningen, seeing Mumford & Sons live in Amsterdam, enjoying music and the sun at Sziget festival, playing long and complex board games in the garden, getting made as the spy, failing to efficiently plant beans, playing one-minute chess in the train, struggling for power in Dungeons & Dragons, climbing up walls upside down in the boulder hall, eating breakfast at bakeries, dining at a fancy restaurant in Delft, watching movies with my girlfriend, going to the night of the discovery in Leiden, watching my friends graduate, and hopefully following suit shortly, are among the events that constitute the time span during which I was working to successfully complete this project.

To every single person that contributed to these events, and to whoever will contribute to those that are yet to come, thank you. As for this thesis specifically, I would like to thank Irina Chiscop for the seemingly endless rounds of detailed and humorous feedback and the proper combination of being my friend and boss at the same time, Karen Aardal and Frank Phillipson for their feedback and experience throughout the project, enabling the submission of two papers, and Roy van Houte for the prolific cooperation that has led to the second part of this work. Regarding the other interns at TNO, I unfortunately have to - involuntarily - thank Stijn Pletinckx, for he alone will take it personally if I were to omit his name from this text.

The German philosopher Arthur Schopenhauer once said that the worst is yet to come, but may I remind you, that so is the best.

J. Mulderij
The Hague, October 2019

Contents

1

Introduction

The first known use of the word “computer” was in 1613, by the English writer R. Braithwait. Back then, the term referred to a human computer, a person who carried out calculations or computations. The word continued to have this meaning until the mid-20th century, when it began to indicate a machine that carries out calculations. Since the development of the Turing machine [76] during the Second World War, the programmable digital electronic computer has taken the human race into a new era, the Information Age. The computational power of the computer has increased exponentially since the 1970’s, following the predictions of G.E. Moore [60]. Technological progress has allowed engineers to place transistors ever closer together, until the effects of quantum physics, such as *tunneling*, brought an end to the exponential improvements.

Instead of interpreting quantum physics merely as an obstacle, a new type of computer has been conceptualized, the quantum computer. The rules that govern physical interactions in a quantum setting allow quantum computing to provide algorithms with a better complexity scaling than their classical counterparts for many naturally arising problems. Exploiting phenomena such as *superposition* and *entanglement*, one can search in a database [32], factor integers [71] or estimate a phase [63] more efficiently than previously possible.

There are two main types of quantum computers, on the one hand the quantum annealer, such as D-Wave [6], and on the other the gated quantum computer, such as IBM Q [13] and Quantum Inspire [67]. The quantum annealer is a computer in which problems can be modeled in terms of a specific Hamiltonian, after which the computer performs the quantum analog of the simulated annealing heuristic [43], which is in turn a digital analog of slowly cooling down metals to enhance certain properties [37]. The quantum annealer does so in order to find the feasible solution to the modeled problem, with the lowest energy, which correponds to an objective function. The biggest limitations in this area are the interaction capabilities between the qubits and the requirements on the form of the provided Hamiltonian. Gated quantum computers are universal and can perform any computation that a classical machine would be able to perform [58, 63]. On a gated quantum computer, a calculation is performed by executing a quantum circuit, consisting of quantum bits (qubits) and quantum gates (hence the name). There are multiple technologies that could potentially form the basis of such a gated quantum computer.

The many advantages of quantum computing come at the price of physical limitations in circuit design. First, coherence times indicate that information on qubits is perturbed or even lost after some time due to a qubit’s interaction with its environment [23]. It is therefore, for a fixed number of qubits, desirable to do calculations with as few gates as possible. A second limitation is posed by nearest neighbor constraints, where two-qubit quantum gates can only be applied to a pair of qubits when the qubits are physically adjacent to each other. This restriction greatly impacts quantum circuit design, since no longer only the function of a circuit matters, but also the manner in which it is implemented. Many proposed methods consider already designed circuits that do not comply with nearest neighbor constraints. In these approaches, SWAP gates, which interchange the location of two adjacent qubits, are inserted into the circuit. The goal herein is to minimize the overhead, i.e., the number of required SWAP gates, to make the whole circuit compliant.

Nearest neighbor constraints have long been a subject of great intersest in a wide variety of fields. First of all, nearest neighbor constraints have been considered in proposals for a range of potential technological

realizations such as ion traps [4, 50, 62], nitrogen-vacancy centers in diamonds [62, 87], quantum dots emitting linear cluster states linked by linear optics [20, 35], laser manipulated quantum dots in a cavity [40] and superconducting qubits [24, 53, 64].

Second, they are also considered in realizations of specific circuits types such as surface codes [78], Shor's algorithm [28], the Quantum Fourier Transform (QFT) [74], circuits for modular multiplication and exponentiation [55], quantum adders on the two-dimensional NTC architecture [16], factoring [66], fault-tolerant circuits [52] and error correction [29].

Third, nearest neighbor constraints have been considered in a range of qubit architectures. In this context, what it means for qubits to be adjacent is dependent on the topology of the architecture. A variety of architectures have been considered: qubits are placed on a linear array in [8, 15, 36, 48, 52, 57, 65, 68, 75, 82, 83], on a two-dimensional grid in [2, 7, 16, 22, 33, 54, 66, 70, 72], on a three-dimensional grid in [27], and, more recently, on the IBM QX architectures in [25, 84, 89, 90].

Until now, the design of quantum circuits consists of manual work in elementary cases and for specific circuits. As the complexity of the algorithms increases, however, manual synthesis will no longer be feasible. When constructing a circuit from scratch, using only the set of elementary gates, even without considering nearest neighbor constraints, one is solving specific instances of the PSPACE-complete Minimum Generator Sequence problem [39], where the group consists of all unitary matrices and the elementary gate operations form the set of generators. Here, one tries to find the shortest sequence of generators to map an input to a given output. A lot of work was done in this area using boolean satisfiability [31], template matching [56, 68] and methods for reversible circuits [3, 81] as all quantum gates perform unitary operations [63].

The early quantum computers have a (very) limited number of qubits [79]. It is possible to connect multiple quantum computers to form a network and do computations together. This is, analogously to current methods in ICT, called *distributed quantum computing* [11, 19]. In such a system we require the network to be able to share both classical and quantum information. If the network is set up correctly, the collection of quantum computers will behave as one big computer [88], and thus greatly increase the possibilities and practical instances that it can be used for.

To act as one big quantum computer, two quantum computers are connected by an entangled pair of qubits. Depending on the topology of the network, we may have a situation where two computers are not connected directly, but indirectly, via other computers in the network. We can apply a method called *entanglement swapping* [45] to create an entangled pair of qubits between these computers. Computers that perform entanglement swapping will need two extra qubits to store and measure information. If we want to perform a calculation on a distributed quantum computer, we have to partition the calculations into parts and assign those parts to the individual quantum computers, in such a way that communication between parts of the calculation is possible and can be done efficiently. The nearest neighbor constraint is one of the important considerations that needs to be taken into account.

In the research area of nearest neighbor compliant quantum circuit design, we recognize four approaches that each consider the problem of minimizing calculation overhead from a different perspective. Global and local reordering split the domain into two, where both can be considered on a single quantum computer, or in the context of distributed quantum computing. The four areas of research are indicated in Table 1.1.

Table 1.1: Four areas of research.

	Local	Global
Single	I	II
Distributed	III	IV

The local reordering problem concerns the minimization of the number of SWAP gates that are inserted into a circuit to make the circuit comply with the nearest neighbor constraints. The local reordering problem allows for any change in the qubit order before each gate, resulting in a vast feasible region, even for small instances. The more general problem of SWAP minimization where qubits are placed on a coupling graph (two qubits can share a gate if their corresponding nodes share an edge) is shown to be NP-Complete [73] via a reduction from the NP-complete Token Swapping problem [9, 42]. The problem we consider, where the graph is a simple path, is widely believed to be NP-complete. This was conjectured in [36], but to the best of the author's knowledge, no formal proof was given as of yet.

To the best of the author's knowledge, the problem of local reordering was thusfar only considered in the context of a single quantum computer (Area I). Many heuristics have been developed in this area, including receding horizon [36, 48, 69, 83], greedy [2, 36], harmony search [2] and OLA on parts of the circuit [65]. Only a few works have dared to approach the problem with exact methods, all of which embody an explicit factorial scaling in the number of variables or processed nodes, either through the use of the adjacent transposition graph [57], exhaustive searches [22, 36] or explicit cost enumeration for each permutation [82]. The exact approaches have delivered small benchmark instances to compare the heuristics' results to. The size of these benchmark instances typically does not exceed circuits of about five qubits and sixteen gates due to the superexponential scaling of the number of variables in the optimization model.

Global reordering determines the initial layout of the qubits such that there are as few as possible SWAP gates required in the remainder of the circuit. In order to elude the micromanagement that local reordering is concerned with, the global reordering problem is generally approximated with the NP-complete [30] problem of Optimal Linear Arrangement (OLA) on the interaction graph of the circuit with edge weights taking the Nearest Neighbor Cost [47]. Here, the gate sequence is either disregarded [70] or encoded in the weights [49]. To the best of the author's knowledge, the global reordering problem is, so far, only considered in the context of a single quantum computer (Area II).

On the interface of distributed quantum computing and nearest neighbor compliance (Areas III & IV), some interesting combinatorial problems naturally emerge. In the combination of local reordering and distributed quantum computing (Area III), qubits are only allowed to interact with their nearest neighbors on the same computer. SWAP operations are once again inserted to make circuits nearest neighbor compliant, and are now also allowed, at a higher cost, between neighboring quantum computers. The global reordering problem is also extended to this context (Area IV), and can be interpreted in two ways. If we are interested in the order of all qubits on all quantum computers, we have *Complete Distributed Global Reordering*. This can be seen as global single reordering with two different cost values for the SWAP gates between qubits on different computers and SWAP gates between qubits on the same computer. We will not consider this variation of the problem in this work since it is only a slight extension when compared to the global reordering problem on a single quantum computer. Next we have, as we will call it, the *Celestial Reordering* problem. In celestial reordering, the goal is to find an initial allocation of the qubits to the computers, such that the number of interactions between computers is minimized. This is done due to the high costs that come with setting up the required entanglement between the computers. The order of the qubits within the computers is not considered. The problem is related to the well known graph partitioning problem.

In the field of quantum circuit design, it is not clear how to modify quantum circuits in a computationally tractable manner, such that they comply with nearest neighbor constraints. Our research question states: "*How can we effectively convert quantum circuits by inserting SWAP gates, such that the converted circuits comply with the nearest neighbor constraints?*" In order to give an encompassing answer to this question, three sub-questions will be answered throughout this thesis:

1. How can we model the nearest neighbor compliance problem such that the size of the model scales polynomially in the input, instead of exponentially, while retaining exact solutions?
2. How does the nearest neighbor compliance problem for two- and three-dimensional architectures relate to the nearest neighbor compliance problem on a linear array?
3. What are the main advantages and disadvantages of exact solution methods for the nearest neighbor compliance problem?

In this work, we attempt to answer these question by considering three qubit architectures. The main focus of this work will be on exact solution methods in research Area I, the local reordering problem on a single quantum computer. The philosophy behind this focus is that small circuits are often incorporated again and again into larger circuits. Obtaining optimal implementations of small circuits can therefore indirectly contribute greatly to the implementation of larger circuits. Furthermore, evaluating benchmark instances with exact solution methods can provide insights into how to develop heuristic methods (that can in turn evaluate large circuits) in the future. The entire contribution of this work can be summarized in fourfold.

First of all, in this work we propose a new model where the Nearest Neighbor Compliance problem (from now on abbreviated as NNC) is formulated as a Single Pair Shortest Path problem on the adjacent transposition graph. Formulating this model and solving it for instances scales very poorly, as $\mathcal{O}(n!)$, where n denotes the

number of qubits in the circuit. The running time, however, experiences only linear scaling in the number of gates. The model might therefore, in instances with very few qubits and many gates, be preferred over other exact methods whose computation times scale worse in the gate count. The approach has a lot in common with the work of [33, 57], where the adjacent transposition graph is used to 1) navigate between permutations of the qubits and 2) count the number of required SWAP gates. We do not evaluate benchmark instances with this model but instead regard it as an intuitive introduction to NNC where qubits are placed on a linear array.

Second, we will, for the first time in the literature, provide an exact integer linear programming (ILP) formulation of the nearest neighborhood compliance problem that does not entail a factorial scaling in the number of qubits, by implicitly counting the number of required SWAP gates at each reordering step (before each quantum gate). The power of the commercial solver CPLEX is used to optimally solve NNC on a linear array for 131 benchmark instances from the *RevLib* library [80]. The considered benchmark instances include the largest circuits to be exactly solved up to this point. They, for example, include the QFT for 10 qubits, and even a circuit with 18 qubits. The evaluation of the larger benchmarks finally allows for heuristics to be compared to exact solutions on larger circuits. Because most heuristics have some strategic flaw that makes them unsuitable for general, large instances, evaluating larger instances exactly might give insights into how to improve current heuristic strategies.

Third, the problem of nearest neighbor compliance will be defined in the setting where qubits are located on two-dimensional and three-dimensional grids instead of on a linear array. In this context, it is significantly harder to count the number of required SWAP gates when moving from one qubit order to another, since there are multiple ways to do it, and it is not trivial to recognize which of the options results in using the fewest SWAP operations. The problem is modeled as a binary integer linear program and solved for a subset of the benchmark instances that were also used in the case of the linear array. In addition to the experimental results, we investigate theoretical results on the comparison between the optimal values in the cases of NNC on a linear array, a two-dimensional grid, and a three-dimensional grid.

Finally, a fruitful collaboration with Roy van Houte, a fellow TU Delft Applied Mathematics student and TNO colleague, has resulted in the discovery of two interesting problems on the interface of nearest neighbor compliance and distributed quantum computing. These related problems, along with the Global Reordering problem, are interesting in their own regard but do not contribute toward the main focus of this work. They will be discussed in a separate part, at the end of the thesis. A case is made for the applicability of the problems in the near future and we provide ILP models for each one of them.

The proposed integer linear programs and their sizes in terms of the number of variables and the number of constraints are shown in Table 1.2.

Table 1.2: The order $\mathcal{O}(\cdot)$ of variables and and constraints of each model is shown. Here, n resembles the number of qubits, m is the number of quantum gates, and M is the number of quantum computers. The grid sizes, where applicable, are indicated by m_1 and m_2. The dimension of the grid is denoted by p.

ILP model sizes for different problems			
Research area	Network/qubit architecture	Variables	Constraints
Area I	Linear array	n^2m	n^2m
	Two-dimensional grid	n^4m	n^3m
	Three-dimensional grid	n^4m	n^3m
Area II	Linear array [70]	n^2	n^2
Area III	Linear array	n^2m+nMm	n^2m+nMm
Area IV	Complete	$nM+n^2$	Mn^2
	General	n^2M^2	n^2M^2
	Linear array	n^2	n^2+M
	Two-dimensional grid	$nM+n^2(m_1+m_2)$	$n^2(m_1+m_2)M$
	General grid	$nM+n^2pM^{(p-1)/p}$	$M+n^2pM^{(p-1)/p}$

This report is divided into two parts. To provide the reader with the required background knowledge on the quantum computing application of the problem, Chapter 2 introduces the topic. In Part I, we discuss the NNC problem in the setting of a single quantum computer, corresponding to research area I. Chapter 3 is concerned with the problem definition of NNC on a linear array, and a literature review. In Chapter 4, the shortest path translation of the problem is presented. Next, the ILP formulation is constructed in Chapter 5.

The problem definition of NNC, where qubits are placed on a grid, is given in Chapter 6. An ILP formulation of NNC on both two- and three-dimensional grids is provided in Chapter 7. The proposed models from Part I will be used to evaluate a number of benchmark instances, and the obtained results are presented in Chapter 8. Each of the other research areas indicated in Table 1.1 are discussed in Part II. The first of the related problems that is discussed in the second part is the Global Reordering problem on a linear array (Area II), in Chapter 9. Then, we make a connection with distributed quantum computing. A problem on the interface of local reordering and distributed quantum computing (Area III) is formulated in Chapter 10. We introduce the Celestial Reordering problem (Area IV), which is defined and discussed for multiple architectures in Chapter 11. Concluding statements and suggestions for future work can be found in Chapter. 12.

This research constitutes a Master Thesis of 42 ECTS for the purpose of obtaining a Master degree in Applied Mathematics at the Delft University of Technology and is combined with a nine month, full time internship at the Dutch research institute TNO.

2

Background for Quantum Computing

The author would like to stress that the topic of the thesis is not the physics involved in quantum computing (abbreviated as QC). To introduce the problem at hand, the topic of QC will be introduced in a brief manner. All the concepts needed for a basic intuition will be addressed in this chapter. In the first section some introductory notation will be introduced. The assumed background knowledge of the reader will be specified, together with references as to where this knowledge could be acquired. A mathematical introduction into QC will be provided in the second section. After that, we discuss physical limitations, quantum gate decomposition and different cost metrics.

2.1. Preliminaries

To understand the basic theory of QC, the reader will be assumed to have knowledge about the most basic concepts from linear algebra and probability theory. Following this chapter, some knowledge about mathematical optimization and complexity theory is assumed. Concepts of (integer) linear programming, problem reduction, complexity classes, and algorithmic design should be familiar to a reader pursuing an in-depth understanding of this work. Information about integer programming can be found in [85].

2.1.1. Graph theory

Here we introduce some of the basic concepts in graph theory that will be used later on, mostly in Chapter 4. For the interested reader, more information on this topic can be found in [21].

A *graph* is a pair $G = (V, E)$ of sets that satisfy $E \subset [V]^2$, so the elements of E form a subset of pairs of elements of V. The elements of V are called *vertices*, or nodes of the graph G. The elements of E are called the *edges* of the graph G. If an edge has a starting node and an end node, we call the edge *directed*. If all of the edges are directed, we say the graph is directed, and we speak of a digraph. Directed edges are also referred to as arcs. When we speak of a digraph, we say $H = (V, A)$, where A is the set of arcs. Usually, a graph is illustrated by drawing a dot for each node, and a line between each two nodes if there is an edge in E that consists of the pair of these nodes. The locations of the nodes and edges is not important. All the information that matters is which pairs of nodes are connected by an edge, and which are not.

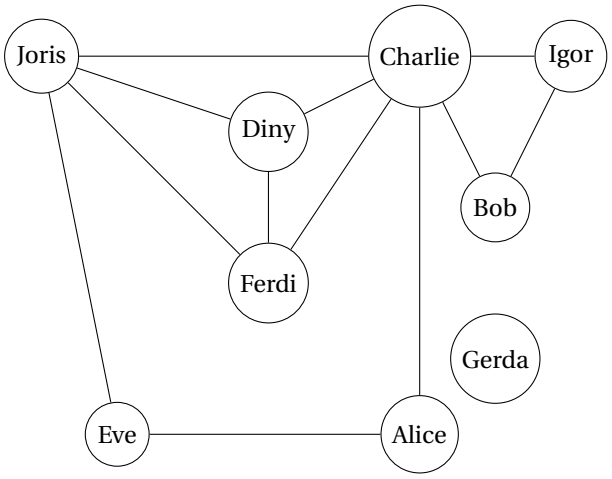


Figure 2.1: An example of a graph on a network of people. The nodes represent the people, the edges could represent friendship relations.

Two vertices $x, y \in V$ are called *adjacent* if there is an edge $\{x, y\} \in E$. For a fixed node x, the number of adjacent pairs of vertices x, y is called the *degree* of x.

2.1.2. Group theory

A few concepts from group theory are introduced in this subsection. A *group* is a set, H, together with an operation $\circ$, called the *group law* of H, that combines two elements a, b from the group to form another element $a \circ b$. The combination $(H, \circ)$ must satisfy four conditions, known as the *group axioms*, to qualify as a group:

1. **Closure** For all elements $a, b \in H$, the result of the operation $a \circ b$ is also an element of H.
2. **Associativity** For all elements $a, b, c \in H$, the equality $(a \circ b) \circ c = a \circ (b \circ c)$ holds.
3. **Identity element** There exists an element $e \in H$ such that for all elements $a \in H$, the equalities $a \circ e = e \circ a = a$ hold. This element e is unique and is called the *identity element*.
4. **Inverse element** For each element $a \in H$, there exists an element $a^\dagger$ such that $a \circ a^\dagger = a^\dagger \circ a = e$, where e is the identity element of H. We call $a^\dagger$ the inverse of a.

The *order* of a group is equal to the number of elements it contains.

Perhaps the most well known example of a group is the *symmetric group* $\mathscr{S}_n$, we will also make use of this group later on in the thesis. The symmetric group on a finite set X is the group whose elements are all bijective functions from X to X, with the group law of function composition. We also refer to the bijective functions as *permutations*. The symmetric group of *degree n* is the symmetric group on the set $X = \{1, 2, \ldots, n\}$. Since there are $n!$ ways to permute the numbers in the set $\{1, 2, \ldots, n\}$, the order of the symmetric group of degree n is equal to $n!$. As for notation, we will let an elements τ_1 and τ_2 of the symmetric group be denoted as

$$\tau_1 = (12)(34)(5) = (12)(34) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{pmatrix} \tag{2.1}$$

and

$$\tau_1 = (124)(35) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 1 & 3 \end{pmatrix}. \tag{2.2}$$

Here, the top row of the matrices is the order of the numbers before the permutation is applied, and the bottom row is the order of the numbers after the permutation is applied. The permutation τ_1 interchanges the numbers 1 and 2 with each other, and it interchanges the numbers 3 and 4 with each other. τ_2 maps 1 to 2, 2 to 4 and 4 back to 1. τ_2 also interchanges the numbers 3 and 5. The function composition $t_1 \circ \tau_2$ is given by

$$\tau_2 \circ \tau_1 = (1)(2354) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 2 & 5 & 3 \end{pmatrix}. \tag{2.3}$$

2.1.3. Linear algebra

We use the complex numbers to represent of qubit states and the interactions they have with each other through quantum gates. We denote a vector

$$\left[\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right] := \alpha\left[\begin{smallmatrix}1\\ 0\end{smallmatrix}\right] + \beta\left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right] = |v\rangle \in \mathbb{C}^2, \tag{2.4}$$

where α and β are both complex numbers.

The complex numbers are equipped with a bilinear map, the inner product. The inner product $\langle\cdot|\cdot\rangle$ of two complex vectors $|v_1\rangle = \left[\begin{smallmatrix}\alpha_1\\ \beta_1\end{smallmatrix}\right]$ and $|v_2\rangle = \left[\begin{smallmatrix}\alpha_2\\ \beta_2\end{smallmatrix}\right]$ is defined as

$$\langle v_1|v_2\rangle = \alpha_1\overline{\alpha_2} + \beta_1\overline{\beta_2}, \tag{2.5}$$

where $\overline{x}$ denotes the complex conjugate of x.

As is usual in linear algebra, we define an orthonormal basis in which we are working through the vectors

$$|0\rangle := \left[\begin{smallmatrix}1\\ 0\end{smallmatrix}\right] \tag{2.6}$$

$$|1\rangle := \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]. \tag{2.7}$$

They form a basis for $\mathbb{C}^2$ and are in the area of QC referred to as the *computational basis*.

A square matrix U is called unitary if its conjugate transpose is also its inverse. Equivalently, U is called unitary if U commutes with its conjugate transpose and the norm of the eigenvalues of U is 1.

The identity matrix of dimension d will always be denoted by I_d, or simply I, if the dimension is implied by the context.

2.2. Components of quantum computing

The terminology that is used in QC is directly related to the terminology that is used classical computing. The quantum analogs of qubits, quantum gates and quantum circuits can be directly related to the original classical concepts of bits, gates and circuits respectively. The theory of QC relies upon the composition of these building blocks, and they will each have their separate introduction. The definitions required to describe the problem formally, and to set up a mathematical model, will be introduced in Chapter 3. This section simply highlights some common topics, useful for intuition and understanding of the context.

2.2.1. The single qubit

The quantum bit (qubit) is the counterpart of the classical bit. Whereas classical bits have a binary value, 0 or 1, a qubit is more nuanced.

The information that qubits hold is represented by a state, denoted by a vector in $\mathbb{C}^2$. Since $|0\rangle$ and $|1\rangle$ form a basis of $\mathbb{C}^2$, the state can be expressed as a combination of the computational basis. Qubits do not have to be in exactly one of the two states, but instead, can be in certain linear combinations of the two basis states.

In general, a state $|\phi\rangle$ of a qubit is displayed as a combination of the computational basis states

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{2.8}$$

We say that the qubit is in (a) *superposition* if it is not in just one of the basis states. If we were to inquire the qubit's information through measuring its state, $|0\rangle$ would be obtained with probability $|\alpha|^2$ and $|1\rangle$ would be obtained with probability $|\beta|^2$. Since probabilities of disjoint events of which the union is the entire probability space have to add up to one, the constraint

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.9}$$

restricts $|\phi\rangle$ to be on the complex unit circle.

2.2.2. Multiple-qubit states

To combine the states of two qubits $|\phi_1\rangle, |\phi_2\rangle \in \mathbb{C}^2$ into one system state $|\psi\rangle \in \mathbb{C}^{2^2}$, a tensor product is used. If two qubits are in states $|\phi_1\rangle = |0\rangle$ and $|\phi_2\rangle = |1\rangle$, the combined state is

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle =: |\phi_1\phi_2\rangle = |01\rangle. \tag{2.10}$$

If we have two qubits that are both in a superposition of the basis states, we can write the combined state as

$$(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes \big(\beta_0|0\rangle + \beta_1|1\rangle\big) = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle. \tag{2.11}$$

Or instead, invoking vector notation, they can also be written as the complex vector

$$\psi = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle =: \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix}. \tag{2.12}$$

Here $\alpha_{ij} = \alpha_i\beta_j$. Note that probability theory tells us that the probabilities of all the states must still add up to one.

In general, an n-qubit state is represented as a complex vector of dimension 2^n. A binary representation is often used to resemble the states. In the two-qubit case the binary representation of the computational basis states represents the binaries from zero up to three, $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$.

If a system state can be written as a single tensor product of single-qubit states, the system state is called pure. The two-qubit state

$$|\psi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{2.13}$$

is a pure state since it can be written as

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{2.14}$$

This is a tensor product of two single qubit states.

If the system state is not a pure state, so if it cannot be expressed as a single tensor product of single-qubit states, the state is called *entangled*. For instance, the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.15}$$

is an entangled state. Intuitively, a state is called entangled if the following holds: if we would measure one qubit's state, it would tell us something about the other qubits' states. In the provided example in Equation 2.15, if the first qubit were to be measured, the value of the second qubit is instantaneously clear as the two qubits must be in the same state.

Imagine two qubits that have entangled quantum states. Now we physically remove them form each other while leaving their states unchanged. Now if one of the qubit's state is measured, the information about the other qubit's state is instantly available (without a delay due to the speed of light). This phenomenon was referred to as "spooky action at a distance" by Einstein [26] and used in [34], where the qubits were separated by as much as 1.3 kilometers!

2.2.3. Quantum gates

Similar to classical computing, where the values of classical bits are adjusted by gates, qubits states are transformed by quantum gates. This is the manner in which computations are done on a quantum computer.

Whereas qubit states are represented by complex vectors $|\phi\rangle$, quantum gates are represented by complex, unitary matrices U. The action of a gate on a qubit state is simply the matrix vector product $U|\phi\rangle$.

Say we have a quantum gate U that acts only on one qubit q_1, in a two-qubit system. Then the action of the gate on the total system is given as the tensor product

$$U_{q_1}|\psi\rangle = U|\phi_1\rangle \otimes I_2|\phi_2\rangle. \tag{2.16}$$

Let us look at some examples of elementary quantum gates. First, there is the Pauli X gate, the quantum analog to the classical NOT gate. Its matrix representation is

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.17}$$

Given a single qubit state $|\phi\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}$, then the X gate has the action

$$X|\phi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_0 \end{bmatrix}. \tag{2.18}$$

It flips the probabilities for finding $|0\rangle$ or $|1\rangle$ when measuring the state.

Other well known examples of single-qubit gates, together with their corresponding matrix representations, are shown in Figure 2.2.

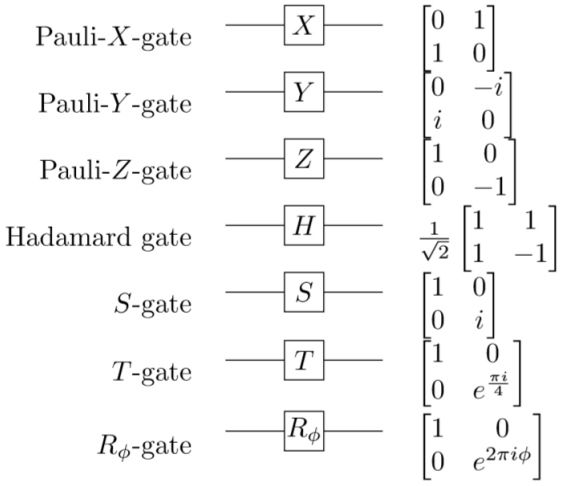


Figure 2.2: These are the elementary quantum gates, together with their depictions and the corresponding matrices. Note that in the last gate, ϕ can be any real number. Figure from [17].

Quantum gates are always reversible ($|\det(U)|^2 = 1$ so U has an inverse). Consequently, there is no quantum analog for the AND, OR and XOR gates. Since quantum gates have an inverse, a gate that has two qubits as an input, must also have two qubits as an output, and their mapping must be both injective and surjective. Fortunately, unitarity is preserved under tensor products.

We will now discuss a few examples and properties of two-qubit gates. The most common example is a controlled gate, where there is one control qubit and one target qubit. If the first qubit is in the state $|1\rangle$, the gate U is applied to the second qubit, otherwise nothing happens. The general expression is

$$C(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & & \\ 0 & 0 & & U \end{bmatrix}.$$

Therefore, if U is the Pauli-X-gate we have what is called a controlled NOT gate, CNOT in short. The CNOT gate is depicted in Figure 2.3.

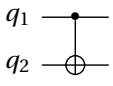


Figure 2.3: Example of a controlled NOT gate. Qubit q_1 is the control qubit and q_2 is the target qubit.

Another common example is the SWAP gate. This gate will be of utmost importance in the rest of this work, so it will be explored in a bit more detail.

The SWAP gate interchanges the states of two qubits. Given two qubits q_1, q_2 with the system state $|\phi_1\phi_2\rangle$, let the matrix

$$U_{\text{SWAP}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

be the matrix corresponding to the SWAP gate. The action of U_{SWAP} on the combined state is

$$U_{\text{SWAP}}|\phi_1\phi_2\rangle = |\phi_2\phi_1\rangle. \tag{2.19}$$

This can be easily checked by investigating the action on the computational basis.

$$\begin{aligned} U_{\text{SWAP}}|00\rangle &= |00\rangle \\ U_{\text{SWAP}}|01\rangle &= |10\rangle \\ U_{\text{SWAP}}|10\rangle &= |01\rangle \\ U_{\text{SWAP}}|11\rangle &= |11\rangle. \end{aligned}$$

The map linearly extends to any superposition of the basis states.

The CNOT gate is used to construct the SWAP gate by placing three CNOT gates consecutively, such as in Figure. 2.4.

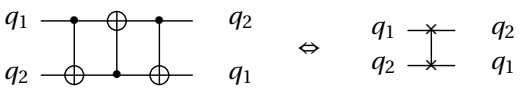


Figure 2.4: A decomposed and composite SWAP gate. The operations are equivalent, the gates interchange the states of two qubits.

2.2.4. Quantum Circuits

A quantum circuit is a cascade of gates acting on an array of qubits. They can be used to calculate all sorts of things, in the same way as classical computers can.

Quantum circuits with n qubits are depicted as n parallel horizontal lines. Each gate has its own symbol or indicative letter to show its function. The gates are places on the horizontal lines from left to right, where the leftmost gate is applied first and the rightmost gate is applied last. Gates acting on more than one qubit connect the corresponding horizontal lines vertically. An example is provided for clarity in Figure 2.5.

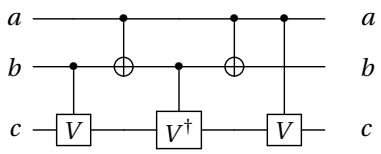


Figure 2.5: Example of a quantum circuit.

In this example, three qubits a, b, c are acted upon by a set of gates. In order, the first gate $C(V)$ is a gate controlled by the qubit b that acts on qubit c. Next, a controlled NOT gate has a as control qubit and b as target qubit. Then, the gate $C(V^\dagger)$ is applied using qubits b and c, etc. This circuit is equivalent to a NOT gate with two control qubits a and b, and target qubit c.

2.3. Physical limitations

In physical implementations of quantum computers, one of the two limiting factors is that qubits are only stable for a very short amount of time [23]. After a while, their state is perturbed by the environment, the

information is lost and the still ongoing calculation becomes useless. It is therefore paramount that the calculations are done as fast as possible. This means that the number of gates should be as small as possible, since it takes time to apply a gate.

The other limitation will be the focus point of this thesis. An issue with quantum gates is that they can only be applied to adjacent qubits. States on qubits can be interchanged, as we saw, using SWAP gates, so there is a way to deal with this problem. However, since there is an incentive to use as few gates as possible, this also imposes a limitation on the number of SWAP gates to be used. It only makes sense to speak of nearest neighbor interaction constraints if the considered gates act on exactly two qubits. In the next section we show how one can decompose gates into two-qubit gates.

To clarify this with an example, the circuit in Figure. 2.5 is made nearest neighbor compliant by inserting a SWAP gate. As a result, all gates now act only on adjacent qubits, while the function of the circuit remains unaltered.

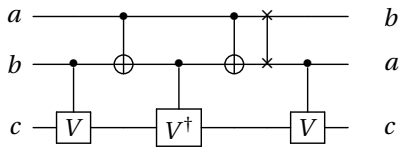


Figure 2.6: Equivalent circuit to Figure 2.5, but now nearest neighbor compliant. A SWAP gate is introduced so that the gates only act on adjacent qubits while leaving the calculation intact.

2.4. Decomposing multi qubit gates

Many circuits make use of composite gates, resembling entire systems themselves. They are often performed on more than two qubits at once, take for example the Quantum Fourier Transform (QFT) which can act on any number of qubits [63]. In order to describe what it means for a gate to act on adjacent qubits, it only makes sense to consider two-qubit gates. To achieve this without changing the function of the circuit, we have to do a modification in two cases.

1. Gates that only act on a single qubit are ignored for the rest of this research. These gates are of no interest in this context because a gate acting on only one qubit always complies with the nearest neighbor interaction constraints.
2. Gates that act on more than two qubits are decomposed into two-qubit gates. The fact that this is always possible is discussed in [63].

The second point can be implemented in a great variety of ways and doing this “optimally" is outside the scope of this work. We therefore make two straightforward design choices: 1) we only consider circuits using multiple-control Toffoli gates, Peres gates and multiple-control Fredkin gates up to a certain size. 2) we always decompose a given circuit in the same way. There is clearly room for improvement here but the search space we consider is large enough as it is. We ignore all single-qubit gates during the modification to a nearest neighbor compliant circuit. The most standard decompositions are presented in the following subsections.

2.4.1. Toffoli gate decomposition

The Toffoli gate is a quantum gate that works on three qubits at the same time. The operation it performs is that of a controlled NOT gate with an extra control qubit. The Toffoli gate's decomposition can be found in [5] in the section “Three-Bit Networks" and is illustrated in Figure 2.7.

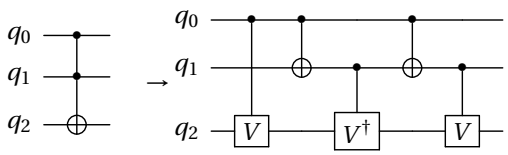


Figure 2.7: The decomposition of the standard Toffoli gate with two control qubits and one target qubit into only two-qubit gates. Here we have $V^2 = X$ where X is the usual Pauli-X gate.

So every Toffoli gate that makes an appearance in the benchmark instances later on, will be replaced by this circuit of two-qubit gates. The Peres gate has a lot in common with the Toffoli gate, it only misses the last gate

in the decomposed circuit. The decomposition can also be found in the circuit “peres_8.real" from RevLib [80]. Notice that these decompositions are not unique, nor are they optimized over. If others use a different gate decomposition method, the instance that is solved differs and the results may vary even if both instances are solved to optimality.

2.4.2. Multiple-control NOT gate decomposition

The decomposition of a three-control Toffoli gate is also shown in [5], in the section “n-Bit Networks". The decomposition of the four-control Toffoli is the direct extension of the previous decompositions. An illustration of the decomposition for the NOT gate with three control qubits is shown in Figure 2.8.

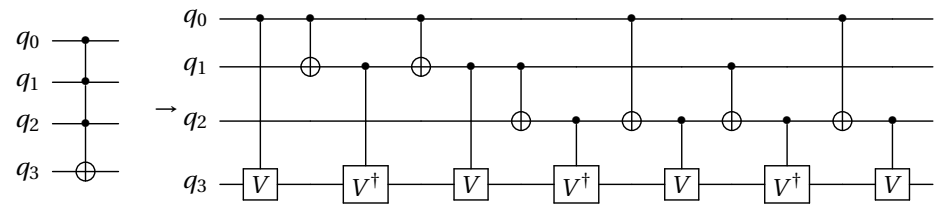


Figure 2.8: The decomposition of a Toffoli gate with three control qubits and one target qubit into only two-qubit gates. Here we have $V^4 = X$ where X is the usual Pauli-X gate.

We remark that the length of the circuit almost doubles for each extra control qubit within the same gate. It is for this reason that the decomposition for the four-control NOT gate is omitted.

2.4.3. Fredkin gate decomposition

The Fredkin gate operates on three qubits as a controlled SWAP operation. It is decomposed as in the circuit “fredkin_5.real", also from RevLib. In Figure 2.9, its decomposition is depicted.

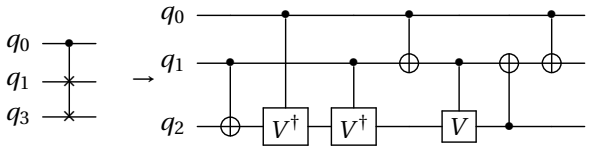


Figure 2.9: The decomposition of the Fredkin gate, or controlled SWAP gate. Here we have $V^2 = X$ where X is the usual Pauli-X gate.

The two-qubit controlled Fredkin gate is decomposed into a controlled-NOT gate, a Toffoli gate and another controlled-NOT gate as shown by [3]. Larger composed gates do not make an appearance in the circuits that are considered in this work.

2.5. Cost metrics

There are many possible implementations of a given calculation. Some circuit implementation require more qubits than others, others require more, or different quantum gates. To distinguish between implementations such that an optimal implementation can be determined, multiple cost metrics exist. Each of these metrics focuses on a different aspect of circuit design, some are more generally applicable and others are centered on specific problems. Most of the cost metrics that we introduce here, are commonly used in the field of quantum circuit design.

1. **Quantum Cost** is a cost metric that describes the total number of elementary quantum gates that a quantum circuit consists of. The elementary gates such as the CNOT gate and the gates listed in Figure 2.2 all have a quantum cost of one. To evaluate the quantum cost of composite gates, one decomposes them into elementary gates and evaluates the sum of the quantum costs of those elementary gates.

2. **The number of SWAP gates** is another metric that is widely used to describe the cost of a quantum circuit. The required number of SWAP gates to make a quantum circuit nearest neighbor compliant are costs that need to be added to the circuit in order to make it physically possible to run the circuit on hardware. SWAP gates are seen as overhead that is needed to do the real calculation. This overhead needs to be kept minimal in size.

3. **Nearest Neighbor Cost** was proposed in [47]. It gives an estimate of how much quantum cost is required, to make a circuit nearest neighbor compliant given an initial qubit configuration. This will be discussed further in Chapter 9.
4. **Cost of entanglement** is a cost metric that is rarely used in the context of nearest neighbor compliance. The cost of setting up entanglement between two qubits in different quantum computers emerges in the topic of distributed quantum computing, where computations are performed using multiple quantum computers. This cost will be further discussed in Chapter 11.

The quantum cost of a circuit is often compared to the quantum cost of the modified circuit where its function remains the same but where all the gates act on adjacent qubits only. Evaluating the extra quantum cost that is needed for such a modification is a more precise way of determining the cost than the other cost metrics. In this precise setting, however, gate decomposition methods are often optimized as well. This is beyond the scope of this research. The nearest neighbor cost is often considered when one is trying to find an optimal initial qubit order. This problem, referred to as Global Reordering, will be discussed in Chapter 9. In this work, we will instead focus on the number of SWAP gates. This is the most common metric and also allows for clear mathematical formulations.

I

Research Area I: Nearest Neighbor Compliance on a Single Quantum Computer

3

Problem Definition: The Linear Array

Part I concerns research area I, as indicated in Table 1.1. In this chapter, the NNC problem will be formalized. To clarify the problem definition, an example is provided. An analogy to a non-quantum setting is made, and we review the theoretical complexity of the problem. A summary of the current literature on the NNC problem on a linear array is included as well.

3.1. Nearest neighbor compliance on a linear array

Due to the limitation that a gate can only be applied to two adjacent qubits, the problem of Nearest Neighbor Compliance (NNC) arises. For the problem of Nearest Neighbor Compliance, the actual operation corresponding to a gate that is used, is not of importance. Only on which qubits the gate acts matters. First, we need to introduce definitions.

Definition 3.1. The coupling graph $H = (V, E)$ of a linear array consists of a set of nodes V and a set of edges E. The set V contains one node for each location. An edge $\{v_1, v_2\} \in E$ is present if the corresponding locations are adjacent (differ by 1).

The coupling graph of a linear array qubit architecture is illustrated in Figure 3.1.

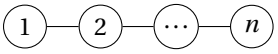


Figure 3.1: The coupling graph for a linear array qubit architecture. There is one node for each location. All the nodes, except for node 1 and node n, have degree 2.

We denote the vector Q of n qubits as the integers $Q = (1, \ldots, n)$. Since the qubits each occupy one physical location on a linear array, the locations are numbered as $L = (1, \ldots, n)$. To keep track of the location of each qubit before every gate, the notion of a qubit order is introduced.

Definition 3.2. Let $\mathscr{S}_n$ be the permutation group. Then a *qubit order* is a permutation $\tau \in \mathscr{S}_n$ of the locations $\tau : L \to L$ denoted by the vector $\tau(L)$. We call τ^t the qubit order before gate g^t.

Example 3.3. In the case where the qubit order τ is given as:

$$\tau = (12)(345) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 5 & 3 \end{pmatrix}, \tag{3.1}$$

would mean that qubit 1 is at position 2, qubit 2 is at position 1, qubit 3 is at position 4, qubit 4 is at position 5, and qubit 5 is at position 3. We say $\tau(i) = j$ if qubit is at position j.

Now that the qubit orders are defined, one requires a way of altering such an order. This is accomplished via the previously mentioned SWAP gates. A SWAP gate interchanges two adjacent locations' qubits.

Definition 3.4. A *SWAP gate* is an adjacent transposition $\tau \in \mathscr{S}_n$ that permutes a qubit order,

$$\tau = (i \quad i+1) \quad \text{for } 1 \leq i < n, \tag{3.2}$$

by interchanging two locations.

It is interesting to know how many SWAP gates one minimally requires to sort a permutation into order (or into any other permutation). There is a metric that tells us exactly how many SWAP gates are required to do so.

Definition 3.5. Given two permutations $\tau_1, \tau_2 \in \mathscr{S}_n$ for some fixed n, then the *Kendall tau* distance between τ_1 and τ_2 is defined as

$$I(\tau_1, \tau_2) := \left|\left\{(i, j) \mid 1 \leq i, j \leq n, \tau_1(i) < \tau_1(j), \tau_2(i) > \tau_2(j)\right\}\right|. \tag{3.3}$$

We also provide the definition of inversions, a concept which will be used hereafter.

Definition 3.6. Given two permutations $\tau_1, \tau_2 \in \mathscr{S}_n$ for some fixed n, then an *inversion* between τ_1 and τ_2 is a pair of elements (i, j) that are pairwise in a different order in τ_1 compared to τ_2.

$$1 \leq i, j \leq n : \tau_1(i) < \tau_1(j), \tau_2(i) > \tau_2(j). \tag{3.4}$$

Note that the Kendall tau metric counts the number of inversions between two permutations. The number of SWAP gates one minimally requires to change one qubit order into another is equal to the Kendall tau distance between the corresponding permutations.

The next theorem tells us that counting the number of inversions is equivalent to counting the number of required SWAP gates to sort one permutation to the other.

Theorem 3.7. *Given a permutation $\tau \in \mathscr{S}_n$ working on the integers $(1, \ldots, n)$, let $I(\tau, e) = \rho$ be the number of inversions in τ with respect to the identity permutation e. Then ρ SWAP operations are needed to sort $(\tau(1), \ldots, \tau(n))$ into ascending order.*

Proof. Note that for $\rho = 0$ no SWAP operations are needed, as $\tau = e$.
Suppose $\rho > 0$ and $\tau \neq e$. We make two statements, both of which we show to be false:

1) Every two integers that are mapped to consecutive integers in the image of τ, are in ascending order, so for all $(\tau(i), \tau(i)+1)$ we have $i < \tau^{-1}(\tau(i)+1)$;

2) All integers are mapped to the image of τ in ascending order, so $i < j$ implies $\tau(i) < \tau(j)$.

Now note that *1)* implies *2)*, and in turn, *2)* implies $\tau = e$. Therefore, statement *1)* is false. So there must exist a pair of integers in the image of τ of which the corresponding inverses are not in ascending order. In other words, there must exist integers $j > k$ such that $\tau(j) + 1 = \tau(k)$. Then we construct τ' where we SWAP the images of j and k such that we obtain $\tau'(i) = \tau(i) + 1$ and $\tau'(\tau^{-1}(\tau(i)+1)) = \tau(i)$.

One SWAP operation was performed and this resulted in the elimination of one inversion, τ was altered to τ' with $I(\tau', e) = \rho - 1$. By the principle of mathematical induction, this completes the proof. □

We remark that the Kendall tau metric is left-invariant, so for any $\tau_1, \tau_2, \tau \in \mathscr{S}_n$ we have

$$I(\tau_1, \tau_2) = I(\tau \circ \tau_1, \tau \circ \tau_2). \tag{3.5}$$

Therefore, if $\tau = \tau_2^{-1}$ in Equation 3.5, we obtain

$$I(\tau_1, \tau_2) = I(\tau_2^{-1} \circ \tau_1, e) \tag{3.6}$$

The result of Theorem 3.7 thus extends to any pair of permutations. This result will be exploited in Chapter 5. To clarify Theorem 3.7, we provide the reader with an example.

Example 3.8. Suppose we are given five qubits $Q = (1,2,3,4,5)$, five locations $L = (1,2,3,4,5)$, and a qubit order $\tau : L \to L$, where

$$\tau = (1234)(5) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 1 & 5 \end{pmatrix} \tag{3.7}$$

The number of inversions between τ and the identity permutation e, is equal to three. Theorem 3.7 states that τ can be changed to e by applying three SWAP operations. The proof argues that there always exists an

inversion such that the two inverted qubits are adjacent in the qubit order. In this case, the qubits 1 and 4 induce an inversion (w.r.t. e) and are also adjacent in the qubit order τ. Therefore, the SWAP gate $s_1 = (12)$ acts on adjacent qubits and interchanges the locations of the qubits. We obtain

$$s_1 \circ \tau = (1)(234)(5) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 4 & 2 & 5 \end{pmatrix}. \tag{3.8}$$

Again, two qubits that induce an inversion are adjacent in the qubit order, this time the SWAP gate $s_2 = (23)$ resolves one inversion. The result is

$$s_2 \circ s_1 \circ \tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 4 & 3 & 5 \end{pmatrix}. \tag{3.9}$$

Finally, the SWAP gate $s_3 = (34)$ interchanges those qubits' locations, that induce the remaining inversion. After applying three SWAP operations, the three inversions have been removed. The resulting qubit order is

$$s_3 \circ s_2 \circ s_1 \circ \tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} = e. \tag{3.10}$$

The nearest neighbor interaction constraints can only be formulated once the concept of quantum gates has been properly introduced in this setting.

Definition 3.9. Let $i, j \in Q$ be two qubits such that $i \neq j$. Let g_{ij} be an unordered pair $g = \{i, j\}$. Then we say that g_{ij} is a *quantum gate*, or simply a *gate*, that acts on qubits i and j. When the specific qubits do not matter in the context, the subscripts might be omitted. When multiple gates are present and their order is important, this will be reflected with a superscript as g^t.

Please note that this definition only allows for quantum gates that act on pairs of qubits, if a "gate" acts on more qubits, or on only one, we do not refer to it as a gate in the sense of Definition 3.9.

To describe an entire quantum circuit, multiple gates are needed, and their order is of importance. To this end, the concept of a gate sequence is introduced.

Definition 3.10. Let $g^1, \ldots, g^m$ be m gates. Let G be the finite sequence of gates $G = (g^1, \ldots, g^m)$, then we say G is a *gate sequence* of size m.

Note that the superscripts are simply used to keep track of the sequence of the gates, they do not resemble powers for example. In this work, for any instance, we assume the gate sequence to be given and fixed. Allowing changes in the gate order when some commutative rules are satisfied, as was done in [33, 38, 57], is beyond the scope of this thesis.

We can now introduce the concept of a quantum circuit more formally.

Definition 3.11. Let Q be the set of qubits and G be a gate sequence. Let Γ be a tuple of the set of qubits and the gate sequence $\Gamma = (Q, G)$. Then we say that Γ is a *quantum circuit*.

At the core of the problem are the nearest neighbor (NN) constraints. These constraints are what make the NNC problem interesting in the first place.

Definition 3.12. Given are a gate g^t_{ij} and a qubit order τ^t before that gate. We say that the gate *complies with the NN constraints* if $\|\tau(i) - \tau(j)\| = 1$, i.e., if the qubits on which the gate acts are adjacent. If, given a qubit order for each gate, all the gates in a quantum circuit's gate sequence comply with the NN constraints, we say that the quantum circuit complies with the NN constraints.

Here, $\|\cdot\|$ simply denotes the absolute difference. Now that all these concepts have been formalized, we can continue with defining the problem of NNC.

NEAREST NEIGHBOR COMPLIANCE PROBLEM (LINEAR ARRAY)

Input: A quantum circuit $\Gamma = (Q, G)$ with $|Q| = n$ qubits and $|G| = m$ gates and an integer $k \in \mathbb{Z}_{\geq 0}$.

Question: Do there exist qubit orders $\tau^t, t \in [m]$, one before each gate of Γ, such that the sum of the Kendall tau distances between consecutive qubit orders satisfies $\sum_{t=1}^{m-1} I(\tau^t, \tau^{t+1}) \leq k$ and such that Γ complies with the NN constraints?

In the minimization version of the problem, which we model in the next section, we seek to find the smallest integer k such that the NNC problem is still answered affirmatively. Considering the problem in this manner, the final qubit order is not required to be equivalent to the initial qubit order. Neither do we allow for changes in the gate order, and we also do not optimize over different ways of decomposing multi-qubit quantum gates. The objective function in the minimization problem simply counts the required number of SWAP gates.

Note that calculating the Kendall tau distance between two permutations can be naively done in $\mathcal{O}(n^2)$ time, following the steps of the bubble sort algorithm [44]. A faster computation of the distance, in $\mathcal{O}(n\sqrt{\log n})$ time, can be found in [14].

We will, however, not be concerned with explicitly listing the Kendall tau distances for all $n!$ permutations. In order to avoid the listing process, the metric should be implicitly calculated in the model. The objective function, variables and constraints that allow us to do so, will be introduced in Chapter 5.

Now, we provide the reader with an example.

Example 3.13. A circuit that does not comply with the nearest neighbor constraints can be modified by inserting SWAP gates to change the qubit order in between gates. Two possible modifications are shown in Figure 6.3. The top circuit has more inserted SWAP gates than the bottom circuit.

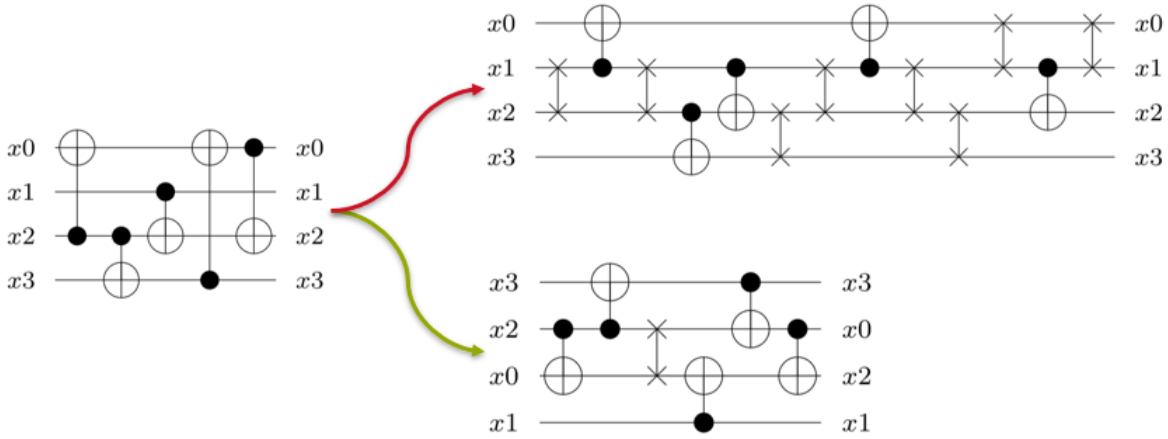


Figure 3.2: Three circuits that perform the same calculation. The circuit on the left does not comply with the NN constraints. The circuits on the right do.

The left circuit does not comply with the nearest neighbor constraints and should be modified in order to do so. The top right circuit is a realization of such a modification, and does comply with the NN constraints. The modification is, however, very poor compared to the modification that the bottom right circuit has undergone. There, only one SWAP gate was inserted instead of the eight SWAP gates used by the top right circuit.

3.1.1. Analogy in a non-quantum setting

The problem can also be envisioned in a non-quantum setting. We consider the following analogy.

Let there be n people, standing side-by-side in a line. There is a piece of paper. On it, a sequence of pairs of names (of the people standing in the line) is written. Starting with the first pair, the two must shake hands, which they can only possibly do when standing next to each other. Then the second pair of people on the list need to shake hands, so on and so forth until every pair in the sequence has shaken hands. Of course, shaking hands with someone is limited by the reach of your arms, so one can only shake hands with direct neighbors.

Now comes the special rule, the people in the line can swap places, but only with their direct neighbors. The goal: finish the list by letting the people swap places as few times as possible.

In this analogy, the people represent the qubits, the line is the one dimensional architecture of the physical implementation of the quantum computer, the list of names represents the list of gates, where every pair of names specifies which two qubits are acted upon by the corresponding gate.

Note that the problem is equivalent to the NNC minimization problem. If we had any feasible solution to the problem above we could translate the place swap of two people to applying a SWAP gate to the two corresponding qubits and obtain a feasible solution to NNC.

3.2. Complexity of NNC and the Token Swapping Problem

Recently, the complexity of a more general form of NNC was studied. The problem of SWAP minimization does not have the qubits on a linear array necessarily, but the coupling between qubits is encoded in a "coupling" graph. Two qubits are considered to be adjacent if their corresponding nodes in the graph share an edge. In this setting, the problem is shown to be NP-complete [73] via a reduction from the NP-complete Token Swapping problem [9, 42].

An application of the Token Swapping problem is the *ladder lottery*, or "Amidakuji", in Japan. The lottery is often used to match children to roles, as per the following example. Imagine an elementary school teacher has to assign two cleaning duties among four students. The teacher might choose a ladder lottery as a means of assigning the tasks in a fair manner. The teacher first draws four vertical lines on a blackboard. Then, several horizontal lines between two consecutive vertical lines, as in Figure 3.3(a). The teacher chooses any two vertical lines, and draws check marks at their bottom ends. The vertical lines are then hidden or covered up. Each student chooses a vertical line and writes their name (or, in this case, a letter) at the top end, as shown in Figure 3.3(b). The vertical lines are then revealed to the students. The students each follow a path down the ladder, starting at the name or letter they chose. The path takes every right and left turn it encounters. Such routes are illustrated with the dotted lines in Figure 3.3(c). The students whose paths end up at a check mark at the bottom are the ones who will perform the cleaning duties. In this case students B and C were chosen.

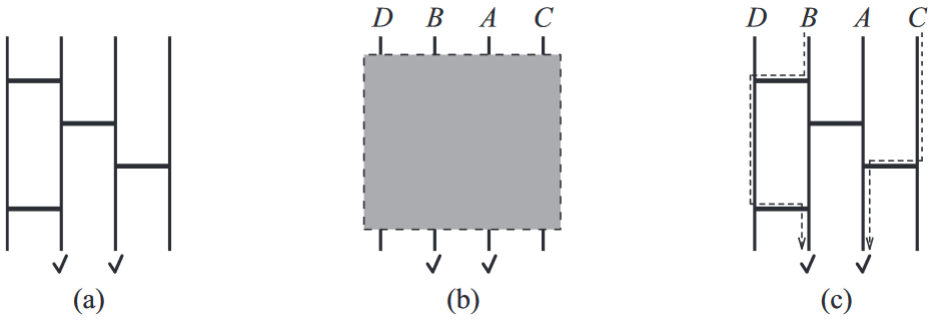


Figure 3.3: How a ladder lottery, or Amidakuji, is used in Japan. Figure from [86].

In the token swapping problem, a graph $G = (V, E)$ is given, with $|V| = n$ nodes. Also n tokens are already placed, each on a distinct node of G. We wish to transform the initial token placement f_0 to a target placement f_t. The tool available to do so, is the swapping of two tokens that are placed on two nodes that are connected by an edge in the graph.

TOKEN SWAPPING PROBLEM

Input:	A graph $G = (V, E)$, with $\|V\| = n$, an initial token placement f_0, a target token placement f_t and an integer k.
Question:	Is it possible to transform the initial token placement f_0 to the target token placement f_t with at most k token swapping operations?

An illustration of the token swapping problem and its relation to the ladder lottery is provided in Figure 3.4.

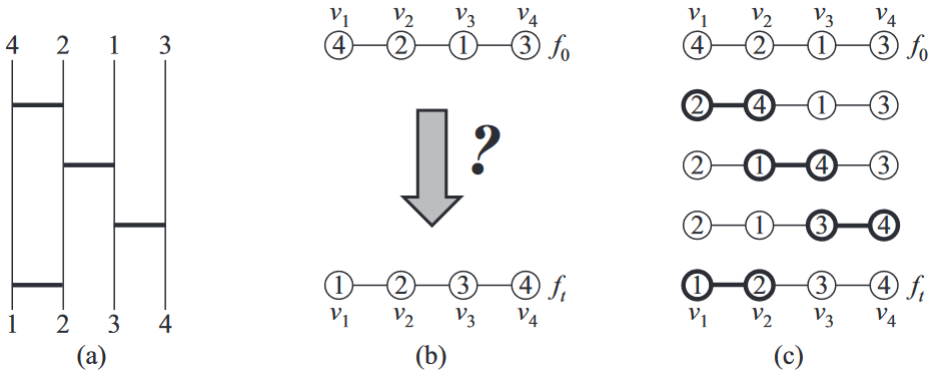


Figure 3.4: How a ladder lottery relates to Token Swapping. Figure from [86].

The token swapping problem is solvable in polynomial time if the graph is a simple path. It is therefore not clear if NNC in this case, where qubits are located on a linear array, is also NP-complete. To the author's best knowledge, there is no formal proof available to show this, it is however widely believed that the NNC problem is indeed NP-complete. This was also conjectured in [36].

Another closely related problem is that of makespan minimization, where the total number of time steps (where one time step resembles the duration of a gate) is minimized. This problem is also NP-complete [10]. Again, the result applies to general coupling graphs and does not trivially extend to planar coupling graphs.

3.3. Literature review

The problem of NNC has been studied extensively in the last decade. Two main categories of coping with the NN constraints have been considered in the literature so far. The first approach is often referred to as global reordering, on which we will elaborate in Chapter 9. The second approach to the problem, the one discussed in depth here, in Part I, is called local reordering. This approach allows for reordering qubits on the gate-level. Here we will discuss the literature on the problem of NNC in the context where qubits are placed on a one-dimensional array. Exact approaches will be discussed first, after which we will discuss some heuristics.

3.3.1. Exact approaches

Only few works have developed exact solution approaches to evaluate the problem of nearest neighbor compliance.

The problem of local reordering has been solved exactly for instances up to ten qubits and five gates using an exhaustive search method [36]. Here, all possibilities are enumerated and the one with the best objective value is returned as the optimal solution. The complexity of searching all qubit orders grows as the factorial of the number of qubits in the considered circuit. The method does therefore not scale well at all as the problem size increases.

Another exact solution method consists of an algorithm making use of Pseudo-Boolean Optimization [82]. Optimal solutions were obtained for circuits up to five qubits and sixteen gates. The boolean satisfiability model introduced by the authors, enumerates the costs for all of the permutations of the qubits and thus scales poorly as the number of qubits increases. Their model does, however, pave the way for the integer linear program (from now on abbreviated as ILP) that we introduce later on, in Chapter 5. They also compare global reordering to local reordering and conclude that local reordering obtains better results due to the bigger search space it allows for.

In [57] an adjacent transposition graph is used. The graph has qubit orders as its nodes, and links two qubit orders with an edge if one can be obtained from the other when applying one SWAP gate. The solution then comes in the form of a path though the adjacent transposition graph. The authors use a gate dependency graph to indicate whether gates can be applied in a different order, an option that will not be considered in this thesis. A breadth first search algorithm is used as a solver. The results indicate that the method finds optimal results quickly in circuits of up to seven qubits and fourteen gates. When the instance size increases to ten qubits, the computation time seems to approach practical limits. For 25 qubits, the solution is no

longer reported. These circuits are already quite large when compared to other exact methods. The problem that is inherent to usage of the adjacent transposition graph, is that the number of nodes scales linearly with the number of qubit orders, which again results in a factorial scaling in the number of qubits.

Another method, from [68], also uses the fact that quantum gates can be interchanged if certain commutation conditions are met, e.g., if two consecutive gates do not act on the same qubits, they can be interchanged in the gate sequence. This results in a yet bigger search space. Compared to a naive method, their results are much better in terms of number of required SWAP gates to transform the quantum circuit into an equivalent nearest neighbor compliant circuit.

3.3.2. Heuristic approaches

A large number of heuristic approaches has been developed for the local reordering approach. The heuristics have been tested on benchmark instances with exact solutions for small circuits and compared to each other in larger instances. The proposed methods include: greedy searches where one qubit is simply moved toward another [36], random permutations are chosen after which the number of SWAP gates are counted [36], a greedy heuristic similar to 2-opt in [1], Sliding Window Search heuristics where only the next fixed number of gates are considered for the reordering [8, 36, 48, 52], and a Fast Sliding Window Search where, additionally, only a restricted number of qubit orders are considered [36]. Another Sliding Window approach uses the concept of the circuit tail [83], in which only part of the tail is used to determine local optimal solutions, which are later on combined to obtain a complete strategy. Other attempts include the use of templates for circuits in the Multiple Control Toffoli (MTC) library [15]. A divide and conquer technique, where the circuit is split into clusters that are considered separately, was developed in [65]. In [75], a solution method is proposed, based on a combination of MTC library templates and the heuristic method of [48].

3.3.3. Our contribution

In the current literature, the lack of an ILP formulation for the local reordering problem is surprising. Multiple models are presented in this work. In Chapter 4 a Single Pair Shortest Path formulation of the problem is presented. This formulation uses the adjacent transposition graph, just as in [57], but traverses multiple instances of it to find optimal solutions without considering the possibility of using a different gate order. The model serves perfectly as an intuitive tool to get a grasp on the problem at hand. In practical use, however, the size of the model quickly scales out of control. In order to solve the problem for instances beyond tiny ones, an ILP model is developed in Chapter 5. The model scales polynomially in size, which is the first one to do so in the current literature. We evaluate a range of benchmark instances by feeding the model to CPLEX's Branch & Bound algorithm. For the first time, instances with up to eighteen qubits and sixteen gates or five qubits and over 100 gates are solved to optimality. Finally, the newly obtained exact solution can serve as a first mean to assess the quality of the existing heuristic solution methods.

4

Shortest Path Formulation

In graph theory, the problem of finding a shortest path between two nodes is a well studied problem. There are many application that require this in practice, the most well known of which is navigation. Imagine you want to travel from point A to point B. The question of how to get there as quickly as possible, is answered by many navigation systems every day. Whenever a package is to be delivered, or when someone wants to go on vacation, one resorts to a navigation system for the shortest route to their destination.

Many different variants of the problem have been studied. In some cases the edges are considered to go only one way or are equipped with weights. The weights often represent distances, travel times or costs. In other variants of shortest path problems, the goal is to find the shortest path between each pair of locations. In this chapter, we consider the first research question: "*How can we model the nearest neighbor compliance problem such that the size of the model scales polynomially in the input, instead of exponentially, while retaining exact solutions?*". We will try to model the NNC problem such that the size of the model scales as a polynomial in the number of qubits and gates, by making use of concepts in graph theory. Throughout this chapter, we attempt to formulate the NNC problem as a Single Pair Shortest Path problem.

4.1. Single Pair Shortest Path

In this case, a simple form of the shortest path problem will suffice. First, we provide the reader with a short introduction into graph theory. Then, a reduction will be done from the NNC problem to the shortest path variant that we discuss.

4.1.1. Formulation of SPSP

In the context of this thesis, we only consider finite graphs $H = (V, E)$, meaning V and E (or A if we speak of a directed graph) have a finite number of elements. We start out by introducing paths.

Definition 4.1. A finite sequence of edges which connects a sequence of vertices is called a *path*. A path is called a directed path, or a *dipath* if all the arcs are directed in the same direction.

A dipath with v_1 as its starting node, and v_2 its end node, is called a $v_1 - v_2$ dipath.

Definition 4.2. A path that does not contain the same vertex twice, i.e., a path without cycles, is called a *simple path*.

The decision version of the Single Pair Shortest Path (SPSP) problem is given as

SINGLE PAIR SHORTEST DIRECTED PATH WITH BINARY WEIGHTS	
Input:	Let $G = (V, A)$ be a digraph with the vertex set V that includes a starting node s and end node t. Let w be a binary weight function on the arcs, $w : A \to \{0,1\}^A$. Let k be a non-negative integer $k \in \mathbb{Z}_{\geq 0}$.
Question:	Does there exist an $s - t$ dipath $P = (a_{s,i}, \cdots, a_{j,t})$ such that $\sum_{a \in P} w_a \leq k$?

In the minimization version of the problem, which is considered in the remainder of this chapter, we seek to find the smallest k for which Problem 4.1.1 is still answered affirmatively. The fact that the path is directed and the weights are binary, will be implicit unless explicitly noted otherwise.

4.1.2. ILP formulation of SPSP

The SPSP problem can also be formulated as an ILP. The decision variables x_a, one for each arc $a \in A$, are introduced. They indicate, for each arc, if the arc is part of the path.

$$x_a = \begin{cases} 1 & \text{if arc } a \text{ is part of the path} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Then the integer linear program corresponding to the SPSP problem is the following:

$$\begin{aligned} \text{minimize} \quad & \sum_{a \in A} w_a x_a \\ \text{subject to} \quad & \sum_{v \in V} x_{i,v} - \sum_{v \in V} x_{v,i} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad \forall i \in V \\ & x_a \in \{0,1\}, \forall a \in A. \end{aligned} \tag{4.2}$$

The corresponding linear programming (LP) relaxation, where the variables are allowed to take on any real value between 0 and 1, is actually integral. This means that every basic optimal solution has each variable equal to 0 or 1.

4.2. Translation of NNC to SPSP

In this section, we introduce the concept of Cayley graphs, we provide a reduction from NNC to SPSP. An (I)LP formulation is therefore available for the NNC problem. The number of variables does, however, increase rapidly as the number of qubits increases. Besides the bad theoretical scaling, this is the first step towards a structured solution method for the NNC problem.

4.2.1. Cayley graphs

A central tool in combinatorial and geometric group theory is the Cayley (color) graph. A Cayley graph captures the structure of an algebraic group in, as the name suggests, a graph.

Definition 4.3. Suppose Z is a group and S is a generating set of Z. The Cayley graph $\zeta = \zeta(Z, S)$ is a colored directed graph, constructed as follows:

1. Each element $z \in Z$ is assigned a vertex.
2. Each generator $s \in S$ is assigned a color c_s.
3. For any $z \in Z$ and $s \in S$, the vertices corresponding to the elements z and zs are joined by a directed edge of color c_s. Thus, the edge set $E(\zeta)$ consists of pairs of the form (z, zs), where s indicates the color of the edge.

The set S is assumed to be symmetric and does not contain the identity element of the group Z.

An example on the dihedral group D_4 is given.

Example 4.4. Consider the dihedral group $Z = D_4$ with its two generators $a, b \in S$. Here, the action of a constitutes a clockwise rotation of $\frac{\pi}{2}$ radians. The action of b is mirroring in the vertical axis. Mapping the structure of D_4 using a Cayley graph results in the graph in Figure 4.1. Notice that b is self-inverse. The blue edges, those corresponding to b, are therefore undirected. Equivalently, the blue edges $\{i, j\} \in E$, could be replaced with two arcs, one going from i to j, and one going from j to i.

Qubit orders are elements of the symmetric group $\mathscr{S}_n$. The adjacent transpositions are a set of generators of the symmetric group, corresponding to the action of SWAP gates. The structure of the symmetric group, with

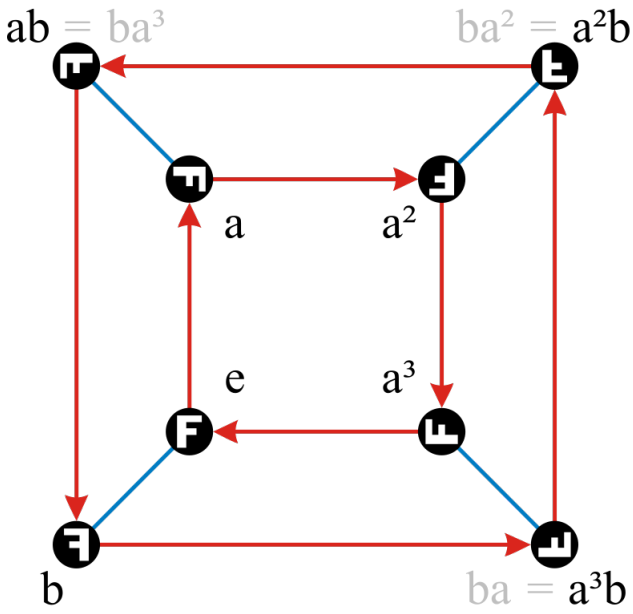


Figure 4.1: The Cayley colored graph corresponding to the dihedral group D_4 acting on the capital letter F. The arcs corresponding to the action of generator a are colored red. The edges corresponding to the action of generator b are colored blue.

as its generator set the set of adjacent transpositions, can also be mapped onto a Cayley graph. We omit the colors and use uncolored Cayley graphs instead.

4.2.2. Graph construction

Translating the NNC problem into a corresponding SPSP formulation, requires the construction of a graph H. The construction contains several steps, which are outlined throughout this section. Let the quantum circuit $\Gamma = (Q, G)$ be given, where $|Q| = n$ and $|G| = m$.

First, all the arrangements the n qubits can be in, are permutations $\tau \in \mathscr{S}_n$ applied on the array $(1, \ldots, n)$. There are $n!$ different permutations of the vector, since $|\mathscr{S}_n| = n!$.

For each gate g^t, a Cayley graph H^t is constructed, that has a node for each qubit order. Thus, H^t contains $n!$ nodes.

Since SWAP gates can be applied before each gate, the qubit order can be altered in each subgraph. SWAP gates can only interchange the position of two adjacent qubits. For each node, an arc to another node is added if and only if the qubit order which that node represents can be obtained by interchanging two adjacent qubits in the qubit order corresponding to the current node. In other words, for each two qubit orders τ_1, τ_2, the corresponding nodes v_1, v_2 are connected if and only if there exists a SWAP gate s such that $s \circ \tau_1 = \tau_2$. These arcs are all given a weight of one. The weights are equal to one, because moving between two nodes in the graph corresponds to altering the qubit order by applying one SWAP gate.

To illustrate the Cayley graph in the case of three qubits, a subgraph H^t is shown in Figure 4.2.

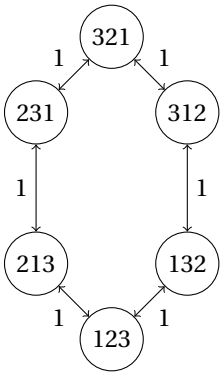


Figure 4.2: A subgraph showing the construction of the subgraph H^t for three qubits. The qubit orders are shown in the nodes. Two nodes are connected via arcs if the qubit order in one node can be obtained from the qubit order in the other node by applying one SWAP gate.

As mentioned before, since the qubit order can be adjusted before every gate, this subgraph is duplicated for every gate. We next describe the manner in which subgraphs are connected to each other. Consider two subgraphs, H^t and H^{t+1}, corresponding to consecutive gates. Only nodes that correspond to qubit orders τ^t, τ^{t+1} for which the gate g^t complies with the NN constraints are connected with an arc from τ^t to τ^{t+1}. The cost on these arcs is equal to zero, since it does not cost any SWAP gates to apply the gate g^t itself.

To illustrate this, another example is given in Figure 4.3. The example uses three qubits and the gate $g^1_{2,3}$.

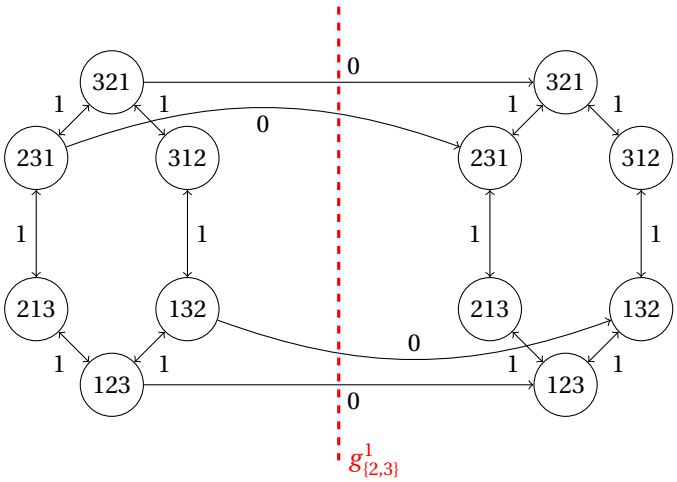


Figure 4.3: The two subgraphs H^t (on the left) and H^{t+1} (on the right) are connected. Only certain qubit orders are allowed to "pass through" the gate. The dashed red line represents the gate.

Finally, the addition of two auxiliary nodes completes the construction. Let s and t be nodes such that s has an arc from it to every node in H^1 and let an arc go from every node in H^{m-1} that would have an arc to H^m by the construction, to t instead. The subgraph H^m is removed from H. All the arcs from s and to t have a cost of zero since the starting configuration can be chosen freely and the final configuration does not have to be the same as the starting one.

4.2.3. Example of graph for a given circuit.

Consider the quantum circuit with three qubits and three gates, shown in Figure 4.4.

The corresponding graph H is shown in Figure 4.5.

From the starting node s, all edges are drawn toward the nodes in H^1 with zero cost, indicating that every initial qubit order is free of costs. Then, arriving in H^1 there is the option of using SWAP gates to move to

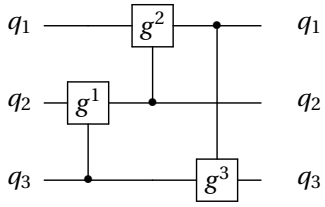


Figure 4.4: This quantum circuit does not comply with the NN constraints.

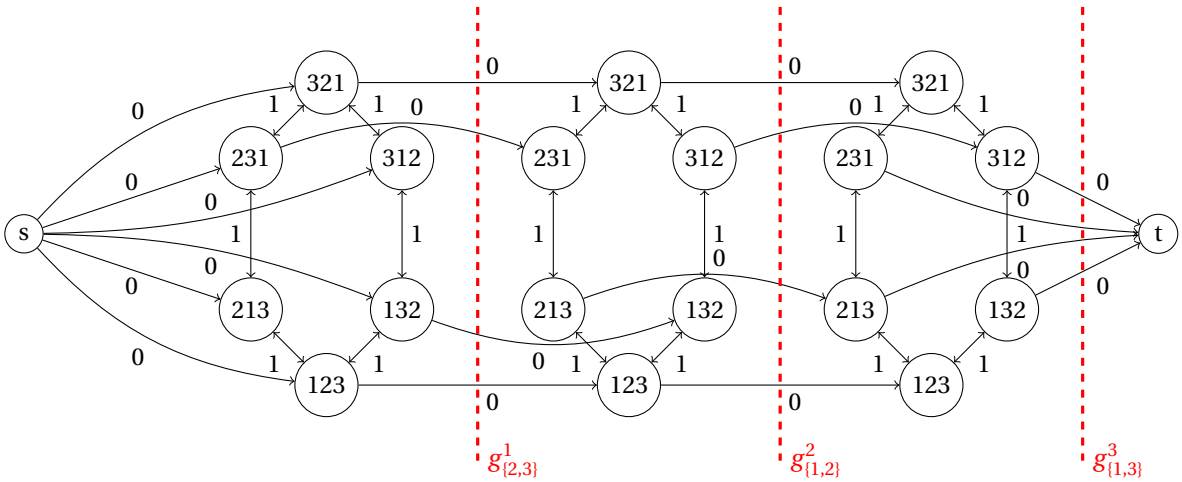


Figure 4.5: Any feasible $s-t$ path in this graph corresponds to a feasible solution to the NNC minimization problem. The total weight of the path is equal to the number of SWAP gates in the corresponding circuit.

a different node in H^1. In some initial qubit orders it is possible to apply gate g_1 and move directly to H^2, without the use of a SWAP gate.

4.2.4. Problem reduction

The anticipated moment has arrived, the NNC problem is now equivalent to solving an SPSP problem from s to t in the constructed graph H. The arcs the path traverses that have a cost of zero, indicate no changes to the qubit order. The edges with a cost of one in the subgraph H^t indicate the use of a SWAP gate to alter the qubit order τ^t, before gate g^t. When traversing the path, the nodes correspond to the current qubit order τ.

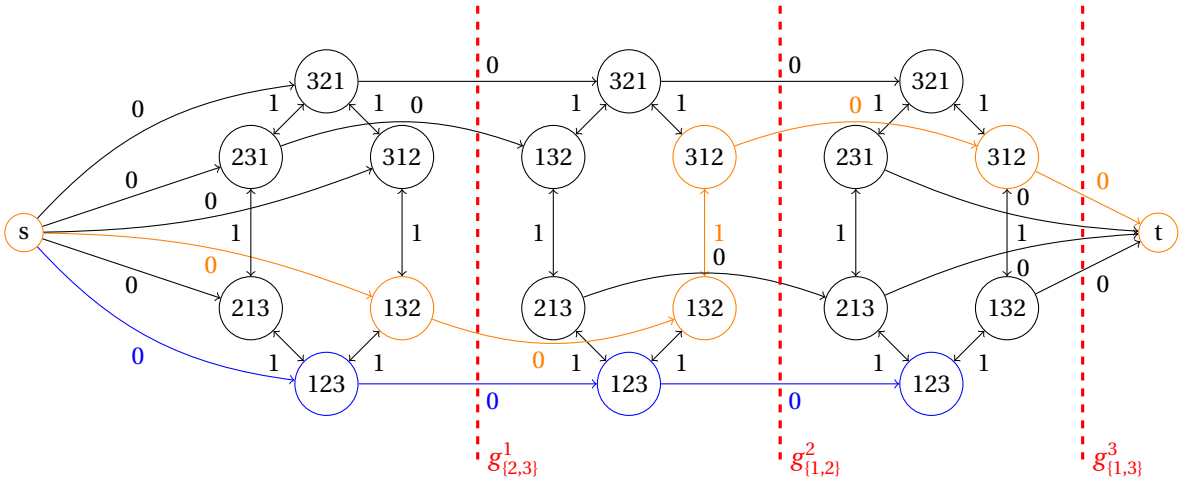


Figure 4.6: Any feasible $s-t$ path in this graph represents a NN compliant circuit. An example of a feasible $s-t$ path is the orange path. It traverses different nodes, corresponding to different qubit orders. A SWAP gate is used before the second gate to ensure the gate complies with the NN constraints.

Two paths are shown in the graph in Figure 4.6. The orange path is a feasible $s-t$ path that uses one SWAP gate to reconfigure the qubits. This way, no gates act on two non-adjacent qubits in the corresponding circuit, so the circuit complies with the NN constraints. The corresponding circuit is also given in Figure 4.7. The blue path corresponds to the circuit in 4.4. Just before the final gate can be applied, a change in the qubit order has to be made in order for the last gate to comply with the NN constraints. Therefore, the blue path cannot continue to the endpoint t without going to a different qubit order first.

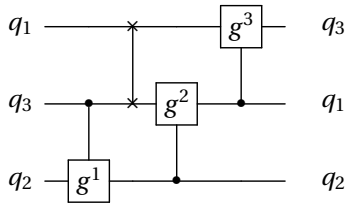


Figure 4.7: This circuit is a NN compliant version of 4.4. Only a single SWAP gate is used.

Next, an example will be presented in Figure 4.8. Here, a feasible, but clearly non-optimal path, is taken in the graph. The result is an excess number of SWAP gates. The same circuit of three qubits and three gates is used as before. The initial qubit order is different, and so is the use of SWAP gates.

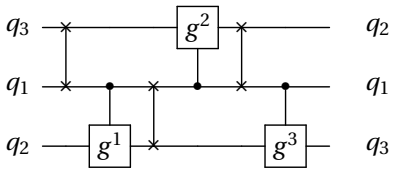


Figure 4.8: This circuit is equivalent to 4.4. Three SWAP gates are used instead of only one.

The $s-t$ path corresponding to the circuit in Figure 4.8 is shown in orange in Figure 4.9.

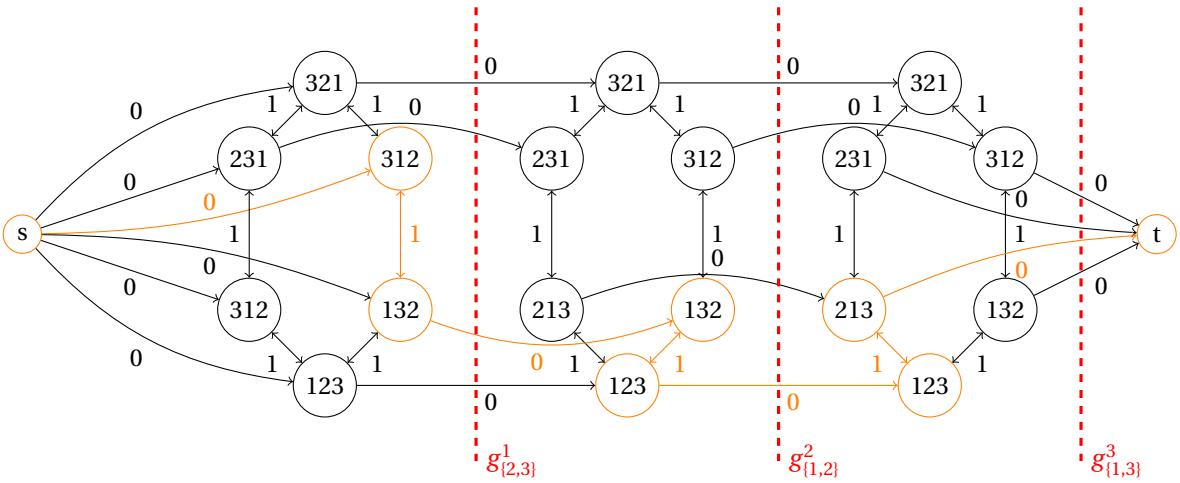


Figure 4.9: The orange path has a total weight of three. The corresponding circuit also contains three SWAP gates.

The number of nodes per subgraph is $n!$, and there is one subgraph for each gate. Thus, also counting the auxiliary nodes, the total number of nodes is equal to $|V| = n!(m-1)+2$. Within each subgraph, the number of edges is equal to $n!(n-1)$, and the number of arcs between subgraphs is $2\cdot(n-1)!$. Together with the arcs from s to the first subgraph, the total number of arcs is $|A| = n! + n!(n-1)m + (n-1)!\cdot 2m$. The size of the SPSP problem scales extremely rapidly with the number of qubits, but only experiences linear scaling in the number of gates.

Example 4.5. To illustrate the rapid scaling to the reader, we show the Cayley graph corresponding to the qubit orders and SWAP gates for only four qubits in Figure 4.10.

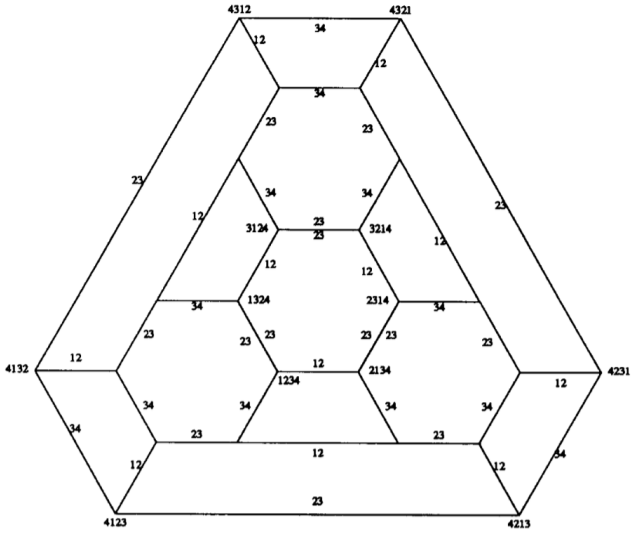


Figure 4.10: The Cayley graph corresponding to $\mathscr{S}_4$ with the adjacent transpositions as generators of the symmetric group. The label $i\,j$ on an edge indicates that the qubits on locations i and j are interchanged by a SWAP gate. The graph has 24 nodes. Figure from [51].

The graph for five qubits already contains $|V| = 120$ nodes.

4.2.5. Reducing the size of the graph

Fortunately, a useful symmetry is present in the problem. The only limitation on any qubit order before a gate, is that the adjacency constraint is satisfied. Notice, that if a qubit order were inverted, so if the first element is interchanged with the last element, the second element with the second to last element, etc., then compliance with the NN constraints is not influenced.

If we were to merge pairs of nodes that represent these inverted configurations, the number of nodes per subgraph H^t, is reduced by a factor of two and becomes $\frac{n!}{2}$. This symmetry was also pointed out in [82], where it was used to halve the number of variables in their SAT formulation of NNC. Notice that the number of arcs is also halved when this symmetry is exploited.

Example 4.6. Consider the example depicted in Figure 4.6. We can reduce the size of the graph significantly by using the symmetry. The nodes 321 and 123, for example, correspond to symmetrical qubit orders. We construct a reduced graph in Figure 4.11.

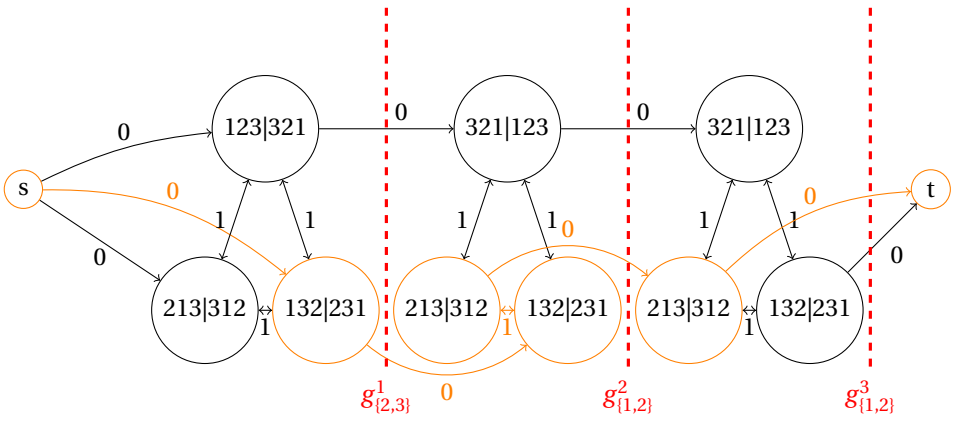


Figure 4.11: The graph H where the symmetry argument is exploited. The number of nodes and the number of arcs have both been halved when compared to the initial construction.

The same optimal solution as in Figure 4.6 is colored orange.

Even though the number of nodes and edges has been halved, their abundance still scales with a factorial in the number of qubits in the corresponding quantum circuit.

4.2.6. Computation time

To have an indication of how the scaling of the model translates to computation times, we perform a short analysis. Since we consider a connected graph, the number of nodes is never bigger than the number of arcs, $|V| \leq |A|$. The weights on the edges are all binary, the problem can therefore be transformed in the following way:

1. Contract all nodes that are connected by an arc a if $w_a = 0$.
2. Ignore the remaining costs altogether.

The last step can be performed since the remaining arcs all have a weight of one. The remaining problem is on an unweighted graph. A Breadth First Search (BFS) algorithm essentially divides the nodes into levels, starting from the starting node s. It visits vertices in order of increasing distance from s by visiting each edge exactly once. Solving a SPSP problem with binary weights can thus be achieved in $\mathcal{O}(|A|)$ time.

Note, that since the NNC problem is (believed to be) NP-Complete and the shortest path problem is solvable in polynomial time, finding a polynomial time problem reduction of NNC to the SPSP problem would mean that either NNC is in the complexity class P, or it would mean that complexity classes P and NP are equal, solving one of the six remaining millennium problems in mathematics. In this case, the reduction time scales as a factorial in the number of qubits. Even though we can solve the SPSP problem in linear time in the number of nodes and arcs, we still acquire a factorial scaling in the computation time due to the number of qubits. We therefore regard this chapter as an intuitive approach to the NNC problem, that clearly illustrates the inherent difficulties in the scaling. We will not use this model to solve the problem for specific instances. For that, we introduce an ILP model in the next chapter.

5

ILP Formulation

In this chapter, we will again consider the research question: "*How can we model the nearest neighbor compliance problem such that the size of the model scales polynomially in the input, instead of exponentially, while retaining exact solutions?*" We attempt to construct a model for the NNC problem that both scales polynomially in size and also allows for the use of exact solution methods. The reason to choose an ILP formulation instead of a SPSP formulation, might seem strange to the reader at first. After all, solving a SPSP problem is easy and solving an ILP is in general not. A motivation and the idea leading up to the change in problem reduction, are presented in the next section. After that, the ILP formulation itself is presented and explained. Next, the scaling of the model is addressed. Finally, we will provide some insight into relaxations of those constraints that require the variables to take integer values. Throughout the chapter, we will assume to be given a quantum circuit consisting of n qubits and m two-qubit quantum gates. A paper on the results of this chapter has been submitted for possible publication [61].

5.1. Motivation and idea

One of the great shortcomings of the SPSP formulation in Chapter 4 is that the number of nodes in the adjacent transposition graph is of order $\mathcal{O}(n!m)$. Even though the SPSP problem is solvable in linear time in the number of variables, it still scales out of control as the corresponding quantum circuit increases in size. Furthermore, when initializing the instance, one has to initialize the variables in some manner. Doing so explicitly, already takes linear time in the number of variables and nodes. We therefore aim to reduce the size of the model while increasing the complexity of solving a problem with it. We illustrate the trade-off in Figure 5.1.

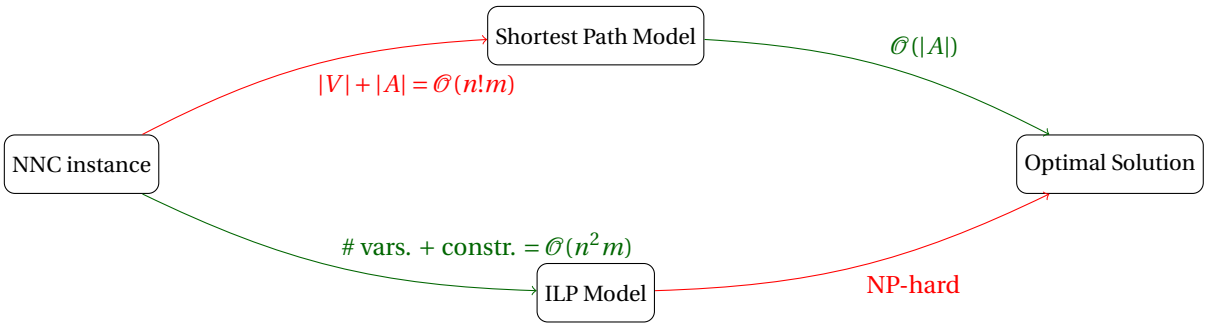


Figure 5.1: The trade-off between a large model with linear solve time and a small model with a bigger solve time. The instance has n qubits and m quantum gates.

5.1.1. Representation of the permutations

The drive behind the $n!$ scaling is the enumeration of all of the different qubit orders. In the SPSP formulation, all permutations were given a node and there are $n!$ different qubit orders for a circuit of n qubits. This

number could be reduced by a factor of two due to a symmetry argument but that does not affect the factorial scaling. Therefore, representing the permutations in some other way than enumeration would be a step in the right direction. At first, an attempt was done to use the traveling salesman problem (TSP) with some, to be specified, weights in order to represent the optimal ordering of qubits before each gate. Note that an ordering of n items can be represented by a vector of length n, instead of listing all possible permutations and keeping track of the applicable one. There are two disadvantages with using n variables to represent the permutation.

1. The variables are no longer binary.
2. To make sure that no two qubits are present at the same position, a "not equal to" constraint has to be added for every pair of qubits, introducing both $\mathcal{O}(n^2)$ new variables and linear constraints.

The qubit order is therefore first represented using a permutation matrix which also requires n^2 variables, however the variables are all binary. Later in this chapter we will still switch to the vector notation because the disadvantages do not weigh up against the advantages of the concise description that the vector notation enables.

5.2. ILP formulation

In this section, we introduce the ILP formulation of the NNC problem. First, the decision variables will be explained, followed by the constraints and the objective function. After these core topics have been discussed, the complete ILP formulation will be presented.

The quantum circuit is cut up into m "time" steps, one for each gate. During each of these steps, any number of SWAP gates can be applied (up to the number of pairs of qubits). The total number of SWAP gates is counted by making use of variables that keep track of the pairwise ordering of the qubits, and variables that keep track of the changes made in the pairwise orderings. The corresponding metric between qubit orders is the Kendall Tau distance between permutations, which was introduced in Definition 3.5. We elaborate on this further, in the paragraphs about the objective function. Throughout the chapter, we use the shorthand notation $[n] := (1, \ldots, n)$.

5.2.1. Decision variables

For every time step $t \in [m]$ the binary decision variable x_{ij}^t is introduced, such that

$$x_{ij}^t = \begin{cases} 1 & \text{if qubit } i \text{ is at position } j \text{ at time } t \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Thus, for every combination of qubit, location and time step, a variable is introduced, indicating if that qubit is at that location, at that time step. In other words, $x_{ij}^t = 1$ if and only if $\tau^t(i) = j$, where τ is the qubit order before gate g^t.

To keep track of the ordering of qubits, binary variables y_{il}^t are introduced for each pair of qubits (i, l) with $i < l$ and each time step $t \in [m]$ such that

$$y_{il}^t = \begin{cases} 1 & \text{if, at time step } t, \text{ qubit } i \text{ comes later in the ordering than qubit } l \text{ at time } t \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Furthermore, binary variables k_{il}^t are introduced to count the number of inversions between two consecutive qubit orders. k_{il}^t is calculated by taking the absolute value of the change of order of two qubits from one time step to the next.

$$k_{il}^t = \left| y_{il}^t - y_{il}^{t+1} \right| = \begin{cases} 1 & \text{if qubit } i \text{ and qubit } l \text{ change pairwise order when going from time } t \text{ to time } t+1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

This is, however, not a linear equation. The manner in which this is taken care of, will be explained in subsection 5.2.3.

5.2.2. Constraints

To make sure that the variables actually correspond to a feasible solution of the NNC problem, some constraints are required. First, two constraints are used to ensure that, at each time step, the variables represent a permutation of the qubits, i.e. every qubit can only be at one position at a time, and every position can only be occupied by a single qubit at a time. The corresponding permutation constraints are

$$\sum_{j \in L} x_{ij}^t = 1 \qquad \forall i \in Q, t \in [m] \tag{5.4}$$

$$\sum_{i \in Q} x_{ij}^t = 1 \qquad \forall j \in L, t \in [m]. \tag{5.5}$$

The adjacency constraints require the qubits that are acted upon by a gate, to be adjacent. This is encoded in the following constraints:

$$x_{ij}^t \leq \begin{cases} x_{l(j-1)}^t + x_{l(j+1)}^t & \text{if } j \in (2, \ldots, n-1) \\ x_{l(j+1)}^t & \text{if } j = 1 \\ x_{l(j-1)}^t & \text{if } j = n \end{cases} \qquad \forall g_{il}^t \in G, j \in L. \tag{5.6}$$

Equation 5.6 is trivially satisfied if $x_{ij}^t = 0$. The constraints require one of the locations $j-1$ and $j+1$, to be occupied with qubit l if location j is occupied by qubit i.

Next, two big-M type constraints are added. The function of these ordering constraints is twofold. On the one hand, the constraints achieve that the y_{il}^t-variables correspond to the values indicated in Equation 5.3. As a result, constraints make sure that the positions of two qubits cannot be equal to each other. On the other hand, the y_{il}^t-variables keep track of the pairwise ordering of the qubits.

$$\sum_{j \in L} j x_{ij}^t - \sum_{j \in L} j x_{lj}^t \leq M y_{il}^t - 1 \qquad \forall i, l \in Q, i < l, t \in [m] \tag{5.7}$$

$$\sum_{j \in L} j x_{lj}^t - \sum_{j \in L} j x_{ij}^t \leq M(1 - y_{il}^t) - 1 \qquad \forall i, l \in Q, i < l, t \in [m]. \tag{5.8}$$

Taking $M = n+1$ suffices as a large enough value. A summation in the Expressions 5.7 and 5.8 sums up to the location of the qubit, i.e.,

$$\sum_{j \in L} j x_{ij}^t = \tau^t(i), \tag{5.9}$$

where τ^t is the qubit order at time t. Note, that one of the two constraints in the Expressions 5.7 and 5.8 is always trivially satisfied. Which one that is, depends on the value of y. This way, y_{il}^t keeps track of which of the two qubits comes first in the qubit order.

Furthermore, all variables are required to take binary binary values.

$$x_{ij}^t \in \{0,1\} \qquad \forall i, j \in Q, t \in [m] \tag{5.10}$$

$$y_{il}^t \in \{0,1\} \qquad \forall i, l \in Q, i < l, t \in [m]. \tag{5.11}$$

$$\tag{5.12}$$

The constraints for the k-variables are introduced alongside the objective function, in the next subsection.

5.2.3. Objective function

With the introduction of the y_{il}^t-variables, we can now keep track of the pairwise ordering of the qubits. As was shown in Theorem 3.7, the number of inversions between two qubit orders is equal to the number of required SWAP gates to change one qubit order into the other. The Kendall tau distance counts the number of inversions by definition.

$$I(\tau_1,\tau_2) = \left|\{(i,j) \mid 1 \le i,j \le n, \tau_1(i) < \tau_1(j), \tau_2(i) > \tau_2(j)\}\right| = \sum_{\substack{i,l\in[n]\\ i<l}} \overline{I}_{i,l}(\tau_1,\tau_2), \tag{5.13}$$

where

$$\overline{I}_{i,l}(\tau_1,\tau_2) = \begin{cases} 1 & \text{if } i \text{ and } l \text{ are in a different order in } \tau_1 \text{ when compared to } \tau_2 \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

The number of inversions between two qubit orders is counted by the y_{il}^t-variables. The equation

$$\overline{I}_{i,l}(\tau_t,\tau_{t+1}) = \left|y_{il}^t - y_{il}^{t+1}\right| \tag{5.15}$$

is a consequence of the construction. Equation claims that the absolute value of the change in pairwise order of two qubits between two qubit orders is equal to one if the qubits changed order, and equal to zero if they did not. Because the absolute value is not a linear function, a common trick is applied.

In general, let $\mathbf{y} \in \mathrm{R}^d$ be a variable vector and $\mathscr{F} \subset \mathrm{R}^m$ the feasible region. Then the program

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \|y\|_1 \\ \text{subject to} \quad & y \in \mathscr{F} \end{aligned} \tag{5.16}$$

constitutes a nonlinear optimization problem. However, a linear optimization problem exists, which has the same optimal value. It requires one extra variable for each dimension of $\mathbf{x}$. The corresponding problem is given in Equation 5.17.

$$\begin{aligned} \underset{\mathbf{y},\mathbf{k}}{\text{minimize}} \quad & \sum_{i=1}^{d} k_i & & \\ \text{subject to} \quad & y_i \le k_i & & \forall i \in [d] \\ & y_i \ge -k_i & & \forall i \in [d] \\ & \mathbf{y} \in \mathscr{F}, \mathbf{k} \in \mathrm{R}^d & & \end{aligned} \tag{5.17}$$

Here, the y_i-variables are upper and lower bounded by the k_i-variables.

The same linearization is applied in our case. Instead of minimizing the sum of all absolute values of differences pairs of consecutive y_{il}-variables, they are bounded by two extra constraints,

$$y_{il}^t - y_{il}^{t+1} \le k_{il}^t \quad \forall i,l \in Q, i < l, t \in [m-1] \tag{5.18}$$

$$y_{il}^t - y_{il}^{t+1} \ge -k_{il}^t \quad \forall i,l \in Q, i < l, t \in [m-1] \tag{5.19}$$

The objective function now becomes

$$\text{minimize} \sum_{t=1}^{m-1} \sum_{\substack{i,l \in Q \\ i<l}} k_{il}^t. \tag{5.20}$$

(5.21)

The inner summation of the variables k is now used to count the number of inversions when changing from the qubit ordering at time t to the one at time $t+1$. The outer summation simply adds the contributions of all time step together.

5.2.4. ILP formulation

Now that all of the variables, constraints and the objective function have been properly introduced, the complete formulation is presented in Equation 5.22. The objective function counts the number of SWAP gates that are required to change each qubit order into the next qubit order. However, the objective function does not unveil the order in which SWAP gates should be applied. Fortunately, this can be easily determined.

$$\begin{aligned}
\text{minimize} \quad & \sum_{\substack{i,l \in Q \\ i<l}} \sum_{t=1}^{m-1} k_{il}^t \\
\text{subject to} \quad & \sum_{j \in L} x_{ij}^t = 1 && \forall i \in Q, t \in [m] \\
& \sum_{i \in Q} x_{ij}^t = 1 && \forall j \in L, t \in [m] \\
& x_{ij}^t \leq \begin{cases} x_{l(j-1)}^t + x_{l(j+1)}^t & \text{if } j \in (2, \ldots, n-1) \\ x_{l(j+1)}^t & \text{if } j = 1 \\ x_{l(j-1)}^t & \text{if } j = n \end{cases} && \forall g_{il}^t \in G, j \in L \\
& \sum_{j \in L} j x_{ij}^t - \sum_{j \in L} j x_{lj}^t \leq M y_{il}^t - 1 && \forall i, l \in Q, i < l, t \in [m] \\
& \sum_{j \in L} j x_{lj}^t - \sum_{j \in L} j x_{ij}^t \leq M(1 - y_{il}^t) - 1 && \forall i, l \in Q, i < l, t \in [m] \\
& y_{il}^t - y_{il}^{t+1} \leq k_{il}^t && \forall i, l \in Q, i < l, t \in [m-1] \\
& y_{il}^t - y_{il}^{t+1} \geq -k_{il}^t && \forall i, l \in Q, i < l, t \in [m-1] \\
& x_{ij}^t \in \{0,1\} && \forall i \in Q, j \in L, t \in [m] \\
& y_{il}^t \in \{0,1\} && \forall i, l \in Q, i < l, t \in [m] \\
& k_{il}^t \in \{0,1\} && \forall i, l \in Q, i < l, t \in [m-1].
\end{aligned} \tag{5.22}$$

5.2.5. Model size

In the ILP model, there are $n^2 m$ variables of type x_{ij}^t, $(n^2-n)m$ variables of type y_{il}^t and $(n^2-n)(m-1)$ variables of type k_{il}^t. This results in a total of $n^2(3m-1) - n(2m-1) = \mathcal{O}(n^2 m)$ variables.

There are $2nm$ constraints to make sure that the x-variables represent a permutation, m gate adjacency constraints, $2(n^2-n)m$ constraints to keep track of the pairwise order of qubits in the qubit order, and $2(n^2-n)(m-1)$ constraints to circumvent the absolute value in the objective function. In total, this results in $4n^2 m - 2n(m-1) + m = \mathcal{O}(n^2 m)$ constraints.

This is a lot better than the $\mathcal{O}(n!m)$ scaling, which we encountered in the SPSP model in Chapter 4.

5.3. Further improvements

To further reduce the number of variables, a concession is made. The x_{ij}^t-variables are replaced by variables that are no longer required to be binary, as was mentioned in Section 5.1.1. Instead, we will use variables that directly indicate the position of a qubit at a certain time step.

$$x_i^t = \sum_{j \in L} j x_{ij}^t = \tau^t(i). \tag{5.23}$$

Example 5.1. As an example, consider a fixed time step t. We have 4 qubits. Their locations are indicated by the matrix

$$X^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \{x_{ij}^t\}, \tag{5.24}$$

where the x_{ij}^t-variables would correspond to the old variables in such a way that the qubits are placed in order of ascending location (q_1, q_4, q_3, q_2). The new variable vector is the vector of locations of the qubits, ordered in ascending qubit number: $(x_1^t = 1, x_2^t = 4, x_3^t = 3, x_4^t = 2)$.

The permutation constraints are no longer required since the old x_{ij}^t variables are no longer in use. The new variables can only take one value each. Furthermore, some adjustment are made to the other constraints. To ensure adjacency of a pair of qubits when a gate is applied to them, the new gate adjacency constraints:

$$x_i^t - x_l^t \leq 1 \qquad \forall g_{il}^t \in G \tag{5.25}$$
$$x_i^t - x_l^t \geq -1 \qquad \forall g_{il}^t \in G, \tag{5.26}$$

are introduced. The ordering constraints already contained the position of the qubits as a function of the old variables. The locations are simply replaced by the new variables. The new ordering constraints[1] become

$$x_i^t - x_l^t \leq M y_{il}^t - 1 \qquad \forall g_{il}^t \in G \tag{5.27}$$
$$x_l^t - x_i^t \geq M(1 - y_{il}^t) - 1 \qquad \forall g_{il}^t \in G. \tag{5.28}$$

Again, $M = n+1$ suffices. The counting constraints remain unchanged. After combining the binary ILP model in the previous section with the new variables, we obtain the following formulation:

$$\begin{aligned}
\text{minimize} \quad & \sum_{\substack{i,j \in Q \\ i<j}} \sum_{t=1}^{m-1} k_{il}^t \\
\text{subject to} \quad & x_i^t - x_l^t \leq 1 && \forall g_{il}^t \in G \\
& x_i^t - x_l^t \geq -1 && \forall g_{il}^t \in G \\
& x_i^t - x_l^t \leq M y_{il}^t - 1 && \forall i, l \in Q, i < l, t \in [m] \\
& x_l^t - x_i^t \leq M(1 - y_{il}^t) - 1 && \forall i, l \in Q, i < l, t \in [m] \\
& y_{il}^t - y_{il}^{t+1} \leq k_{il}^t && \forall i, l \in Q, i < l, t \in [m-1] \\
& y_{il}^t - y_{il}^{t+1} \geq -k_{il}^t && \forall i, l \in Q, i < l, t \in [m-1] \\
& x_i^t \in \{0, \ldots, n-1\} && \forall i \in Q, t \in [m] \\
& y_{il}^t \in \{0,1\} && \forall i, l \in Q, i < l, t \in [m] \\
& k_{il}^t \in \{0,1\} && \forall i, l \in Q, i < l, t \in [m-1].
\end{aligned} \tag{5.29}$$

The number of variables in the model is equal to

$$\#\text{variables} = n^2 m - \frac{n^2 - n}{2} \tag{5.30}$$

[1] Note that these constraints are similar to the Miller-Tucker-Zemlin constraints for the Traveling Salesman Problem [59].

and the number of constraints in the ILP is equal to

$$\#\text{constraints} = 2(n^2 - n)m - n^2 + n + 2m, \tag{5.31}$$

which are both polynomial in the number of qubits and in the number of gates. In other exact solution methods, the size of the model already scales exponentially with the number of qubits [57, 82]. The ILP is harder to solve than the SPSP problem, but we have eluded the factorial scaling inherent to initializing the SPSP problem. The new scaling in the number of variables is quadratic in the number of qubits and linear in the number of gates.

5.4. Relaxations

In order to find lower bounds or to improve running times, it is common practice to relax some or all of the variables to assume not only integer, but also continuous values within the outer limits of their respective domains.

5.4.1. Mixed Integer Linear Program

In relation to the ILP formulation, we state the following theorem.

Proposition 5.2. Relaxing the x_i^t-variables and the k_{il}^t-variables in Equation 5.29 to take continuous variables does not change the objective value of the optimal solution. Thus, there still exists an optimal solution with integer valued variables.

Proof. The x-variables must take values that are pairwise separated from each other by at least 1 due to constraints Equations 5.27 and 5.28. There are n variables that all have to take a value in a connected interval of length n, all spaced at least 1 from each other. This can only be done if the x's are all integer and all integer values are taken. The k-variables are constrained by Equations 5.18 and 5.19. Since the y-variables are binary, their difference is also binary (or -1, in which case $k = 0$ is allowed and optimal). Since we are minimizing over the k-variables, their value will always equal the smallest possible allowed value by the constraints, which is always integer. □

For this Mixed Integer Linear Program (MILP), the solver CPLEX has improved running times in practice. The optimal value is still the same as that of the exact solution to the original problem.

5.4.2. LP relaxation

Understandibly, one could be tempted to relax the y_{il}^t-variables to assume continuous values as well. If all variables are relaxed to take continuous values, we speak of the Linear Programming (LP) relaxation of the problem. This method is often applied in order to obtain a lower bound for the original problem, since linear programming problems are polynomially solvable. In this case, our experience with the LP relaxation tells us that optimal values always provide trivial lower bounds of zero SWAP gates. This occurs as a result of the relaxation of the restriction on the pairwise ordering of the qubits. A similar phenomenon occurs in the Market Split problem [18], where one looks for a binary solution vector to a linear system of equations. If the solution is no longer required to assume integer values, the obtained answer contains almost no information about the problem in the non-relaxed setting.

6

Problem Definition: The Grid

In this chapter, we introduce the NNC problem in the context where we no longer take a linear array as the qubit architecture one can embed a circuit on, but instead we turn to two- and three-dimensional grids. In this chapter and in the next one, we investigate the third research question: "How does the nearest neighbor compliance problem for two- and three-dimensional architectures relate to the nearest neighbor compliance problem on a linear array?"

The reason for the change of qubit architecture is increase of the degree of the qubits in the coupling graph of the grid structures. Intuitively, the required number of SWAP gates will be less when qubits have more neighbors to interact with freely. We prove that this intuition is indeed correct.

First, we introduce a few definitions in order to formulate the problem formally in the settings of two- and three-dimensional grid architectures. Then, we remark on observations and we state four theorems that relate the optimal objective value of the NNC problem on a linear array with those of the NNC problem on the grids. Finally, we discuss the literature that already addresses the NNC problem in the context of grid architectures.

6.1. Nearest Neighbor Compliance on a two-dimensional grid

The new grid architectures extend the meaning of nearest neighbors to the higher-dimensional case. Qubits can, in this setting, interact with neighbors in multiple directions. Therefore, the number of connections each qubit has, increases.

Some definitions are introduced. Denote the set Q of n qubits as the set of integers $Q = \{1, \ldots, n\}$. The qubits all have one physical location in a two-dimensional $n_1 \times n_2$ grid of at least n locations, the locations are given coordinates $L = ((1,1), \ldots, (n_1, n_2))$. If the number of qubits is not divisible by n_1 and n_2, auxiliary qubits are added to fill up the grid. The resulting set of qubits is denoted by Q^*. For example, if we have a set of three qubits $Q = \{1,2,3\}$ and a 2×2 grid, a fourth qubit is added to fill up the remaining location. We then obtain $Q^* = \{1,2,3,4\}$. Note that there will be no gates applied to these auxiliary qubits. We start with the coupling graph in this setting.

The definition of a qubit order in this context is

Definition 6.1. A *qubit order* $\tau : L \to L$, $\tau \in \mathscr{S}_n$ is a permutation on the locations. We call τ^t the qubit order before gate g^t.

Changing a qubit order is now possible in more ways since qubits have more neighbors in a grid as opposed to a linear array. The SWAP gate can still interchange two adjacent qubits.

Definition 6.2. A *SWAP gate* is a permutation $\tau \in \mathscr{S}_n$ on the locations, $\tau : L \to L$, that has the form

$$\tau = (i \quad j), \quad i, j \in L \tag{6.1}$$

such that $\|i - j\|_1 = 1$. Here, $\|\cdot\|_1$ denotes the usual 1-norm.

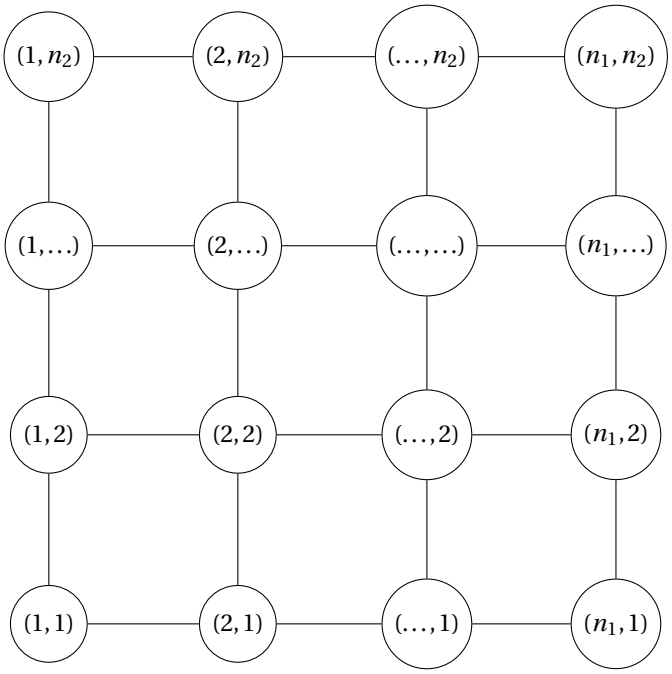


Figure 6.1: The coupling graph for a two-dimensional grid qubit architecture. There is one node for each location. All the nodes, except for those in a corner or on the edge of the architecture, have degree 4.

Notice that this exactly coincides with the adjacency of the coupling graph in Figure 6.1. The definitions of quantum gates, gate sequences, quantum circuits and nearest neighbor compliance for gates and circuits are exactly the same as in Chapter 3. In inherent difficulty of counting the number of SWAP gates that are needed to change one qubit order into another in the setting of a two-dimensional grid, is the lack of an easily calculable Kendall tau-like metric. For now, we will simply call $J(\tau_1, \tau_2)$ the number of SWAP gates that are needed to change τ_1 to τ_2, without providing a manner in which to calculate J. We remark that J is a metric on the qubit orders.

Now that all these concepts have been formalized, we can continue with defining the problem of NNC.

NEAREST NEIGHBOR COMPLIANCE PROBLEM (2D GRID)

Input:	A quantum circuit $\Gamma = (Q, G)$ with $\lvert Q\rvert = n$ qubits and $\lvert G\rvert = m$ gates, locations L on a $n_1 \times n_2$ grid, and an integer $k \in \mathbb{Z}_{\geq 0}$.
Question:	Do there exist qubit orders τ^t, $t \in [m]$, one before each gate of QC, such that the sum of the required number of SWAP gates J between consecutive qubit orders satisfies $\sum_{t=1}^{m-1} J(\tau^t, \tau^{t+1}) \leq k$ and such that the quantum circuit complies with the NN constraints?

In the minimization version of the problem, which we model in the next Chapter, we seek to find the smallest integer k such that Problem 1 is still answered affirmatively. Considering the problem in this way, we still do not require the qubits to end up in the same qubit order as they started out in, nor do we allow for changes in the gate sequence.

We shed some light on the newly defined setting, by making use of the following example:

Example 6.3. To see the translation of a circuit embedded on a linear array to a two dimensional grid, we use the following illustrations. First, the circuit that does not comply with the nearest neighbor constraints, in both architectures.

The circuits require the insertion of SWAP gates in order to comply with the NN constraints. Next, we present a feasible solution to the NNC problem for both architectures. Note that these solutions are not optimal, the

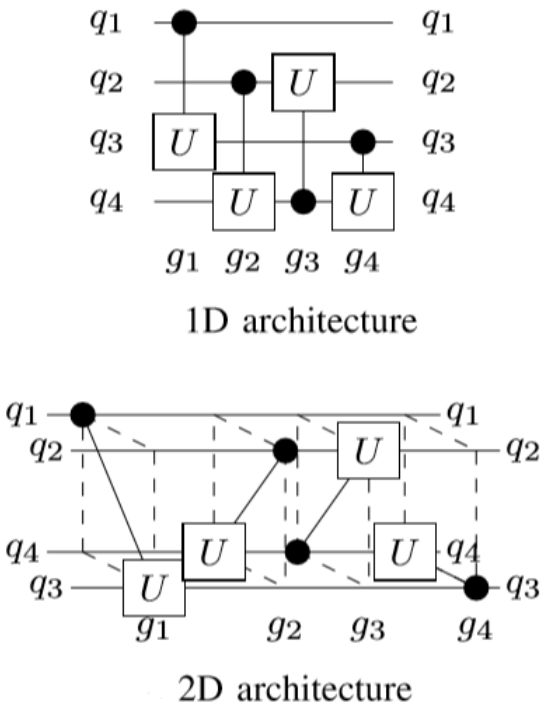


Figure 6.2: A quantum circuit consisting of four qubits and four gates. The gates do not comply with the nearest neighbor constraints. The circuit is embedded on a linear array (top) and on a two-dimensional grid (bottom).

qubit order can still be adjusted such that the first SWAP gate, g_1, can be left out.

6.2. Nearest Neighbor Compliance on a three-dimensional grid

Only a few changes are required to modify the problem such that it applies to the three-dimensional grid architecture. The qubits now all have one physical location in a three-dimensional $n_1 \times n_2 \times n_3$ of at least n locations which are given the coordinates $L = ((1,1,1),\ldots,(n_1,n_2,n_3))$. Once again, auxiliary qubits are added to fill the grid if $n_1 n_2 n_3 \neq n$. The definitions in the previous section naturally extend to the case of the three-dimensional grid.

NEAREST NEIGHBOR COMPLIANCE PROBLEM (3D GRID)

Input:	A quantum circuit $\Gamma = (Q, G)$ with $\|Q\| = n$ qubits and $\|G\| = m$ gates, locations on a $n_1 \times n_2 \times n_3$ grid, and an integer $k \in \mathbb{Z}_{\geq 0}$.
Question:	Do there exist qubit orders τ^t, $t \in [m]$, one before each gate of QC, such that the sum of the required number of SWAP gates J between consecutive qubit orders satisfies $\sum_{t=1}^{m-1} J(\tau^t, \tau^{t+1}) \leq k$ and such that the quantum circuit complies with the NN constraints?

6.3. Observations and claims

In this section we would like to point out some observations regarding the NNC problem. We relate the problems for architectures to each other, and provide the reader with some insights. Let I be an instance of the NNC problem and let $OPT_{1D}(I)$, $OPT_{2D}(I)$ and $OPT_{3D}(I)$ be the optimal objective value to the corresponding minimization version of the NNC problem in the setting of the linear array, the 2D grid and the 3D grid, respectively.

Intuitively, it seems like an optimal embedding of the qubits on a 2D or 3D grid should require less (or at least not more) SWAP operations than on a linear array. To show this is actually the case for any grid sizes and any

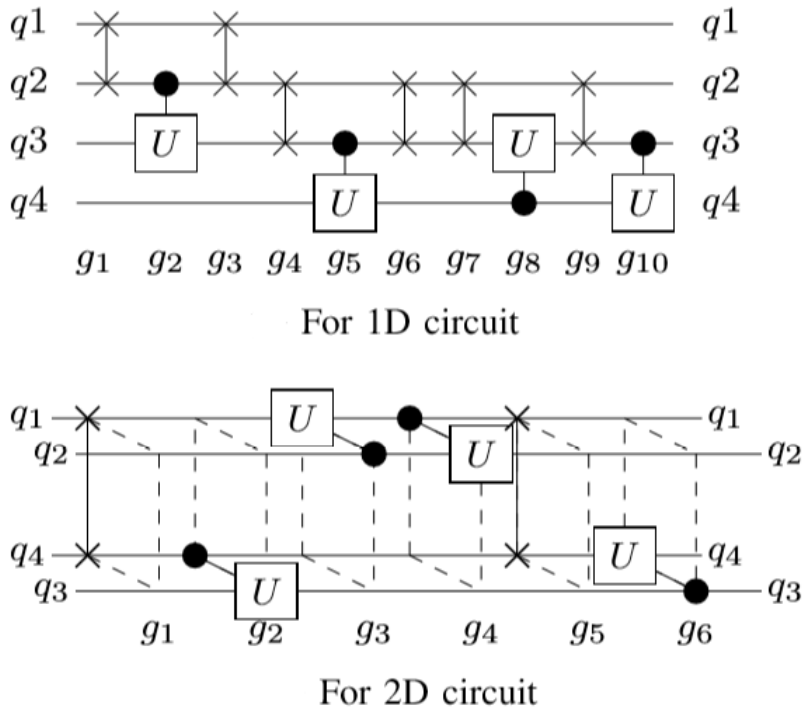


Figure 6.3: A feasible solution to the instance in Figure 6.2 for both architectures. The embedding on the linear array (top) has ten gates, of which six are SWAP gates. The embedding on the two-dimensional grid has six gates of which two are SWAP gates. Neither of the implementations of the circuit correspond to optimal solutions of the corresponding NNC problems.

quantum circuit, we prove the following theorem.

Theorem 6.4. *For any fixed instance I of the NNC minimization problem, with any dimensions of the 2D grid, the optimal value of the NNC problem on a linear array is an upper bound for the optimal value of the NNC problem on a two-dimensional grid,*

$$OPT_{1D}(I) \geq OPT_{2D}(I). \tag{6.2}$$

Proof. We show that every feasible solution of the NNC problem on a linear array can be mapped to a feasible solution of the NNC problem on a 2D grid and that the solution of the 2D grid has at most the same objective value as the solution the the problem on the linear array. First we make the observation that a qubit order on a linear array can be embedded onto the 2D grid in a snake-like pattern. For an $n_1 \times n_2$ grid where $n_1 n_2 \geq n$, let $f : L_{1D} \rightarrow L_{2D}$ be a mapping from the locations on the linear array to those on the 2D grid. We construct f as

$$f : x \mapsto \begin{cases} \left((x-1) \mod n_1) + 1, \left\lceil \frac{x}{n_1} \right\rceil \right) & \text{if } \lceil \frac{x}{n_1} \rceil \text{ is odd} \\ \left(n_1 - ((x-1) \mod n_1), \left\lceil \frac{x}{n_1} \right\rceil \right) & \text{if } \lceil \frac{x}{n_1} \rceil \text{ is even.} \end{cases} \tag{6.3}$$

Here mod denotes the modulo function. Figure 7.1 clarifies the mapping. For any two adjacent locations $x-1, x \in L_{1D}$ in the linear array, the reader can check that $\|f(x-1)-f(x)\|_1 = 1$, so the corresponding locations on the 2D grid are also adjacent. Let x_1 be a sequence of qubit orders for the NNC problem on the linear array. For every qubit order τ^t of x_1, construct a qubit order $f(\tau^t)$ where f is applied on each location. Note that this results in a sequence of qubit orders x_2 on the 2D grid. Also note that x_2 is a feasible solution to the NNC minimization problem on the 2D grid.

We pause for the following intermediate result

Lemma 6.5. Let τ_1, τ_2 be two qubit orders as defined in Chapter 3. Let f be defined as in Equation 6.3. For any qubit order τ in the linear array setting, let $f(\tau)$ be a qubit order in the 2D grid setting by applying f to

each location. The number of required SWAP gates to change τ_1 to τ_2 is at least as much as the number of SWAP gates required to change $f(\tau_1)$ to $f(\tau_2)$, i.e.

$$I(\tau_1, \tau_2) \geq J(f(\tau_1), f(\tau_2)). \tag{6.4}$$

Proof. Let us be given a minimal sequence of SWAP operations in the setting of the linear array which transforms τ_1 into τ_2, we generate a sequence of SWAP operations for the 2D grid that transforms $f(\tau_1)$ into $f(\tau_2)$. For each SWAP operation in the sequence that exchanges the qubits on locations l_1, l_2, apply a SWAP operation to the qubits located at $f(l_1), f(l_2)$ in the 2D grid setting. This sequence of SWAP operations transforms $f(\tau_1)$ into $f(\tau_2)$ and uses the same number of SWAP operations as in the case for the linear array. □

Continuing, by the lemma we find that the number of required SWAP operations on the linear array is at least as much as the number of required SWAP gates in the 2D grid setting. This in turn means that for every feasible solution of the NNC minimization problem on the linear array, there exists a feasible solution with at most the same objective value for the NNC minimization problem on the 2D grid. □

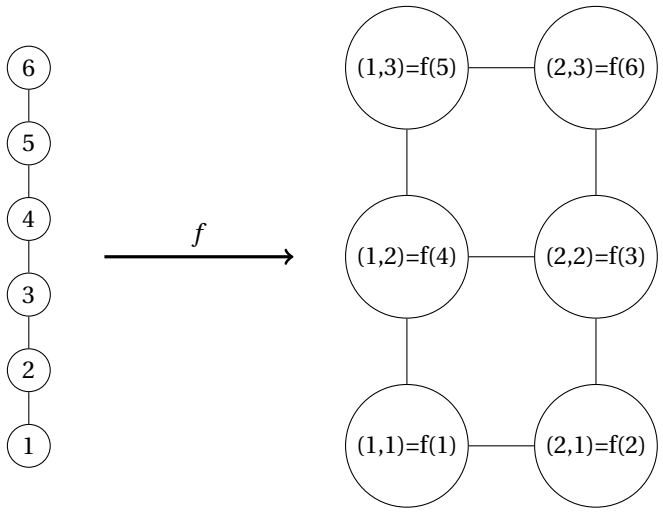


Figure 6.4: An illustration of the mapping f that relates every location on the linear array on the left to a location on the 2D grid on the right. Pairs of adjacent locations in the linear array are mapped onto adjacent locations in the 2D grid. In this case $n = 6$ locations are present, a 2×3 grid is used.

Theorem 6.6. *For a fixed instance I of the NNC problem, for all possible dimensions of the 3D grid, the optimal value of the NNC problem on a linear array is an upper bound for the optimal value of the NNC problem on a three-dimensional grid,*

$$OPT_{1D}(I) \geq OPT_{3D}(I). \tag{6.5}$$

The proof is analogous to the proof of Theorem 6.4 where the snake-like mapping is now extended to a 3D grid. Explicitly giving the mapping would not result in an increased understanding of the proof, the entire proof is therefore omitted. Solving the NNC problem in the setting of a linear array qubit architecture thus provides a feasible solution to the NNC problem for the two-dimensional and three-dimensional cases. In the two-dimensional case, the mapping is given in the proof of Theorem 6.4. For the three-dimensional case, the mapping extends naturally from the two-dimensional case.

Theorem 6.7. *For a fixed instance I of the NNC problem, if at least one of the dimensions of the grids of the two-dimensional grid corresponds with one of the dimensions of the three-dimensional grid, then the optimal value of the NNC problem on a two-dimensional grid is an upper bound for the optimal value of the NNC problem on a three-dimensional grid. So for grids $n_1 \times n_2$ and $N_1 \times N_2 \times N_3$,*

$$\exists (i, j) \in \{1,2\} \times \{1,2,3\} \text{ s.t. } n_i = N_j \implies OPT_{2D}(I) \geq OPT_{3D}(I). \tag{6.6}$$

Once again, the proof is analogous to that of Theorem 6.4, where now the dimension in which the grids have the same size extend the 1D and 2D grids to the 2D and 3D grids respectively.

Theorem 6.8. *Let I be an instance of the NNC problem with $n = 3$ qubits. The optimal value of the NNC problem on a linear array is then equal to that of the NNC problem on any grid.*

Proof. Notice that we are in either of the following two disjoint scenarios:

1. The grid is a linear array.
2. The grid contains a 2×2 grid of locations.

If the grid is a linear array we are done. If the grid contains a 2×2 grid, we show that an embedding of the qubits on the 2×2 grid is no better than the embedding on the linear array. It is clear that if qubits are placed further away from each other throughout the circuit, a solution cannot get any better than the embedding on the 2×2 grid.

From each qubit order on the 2×2 grid, two of the three pairs $\{\{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}\}$ of qubits are adjacent. Using one SWAP gate, every other combination of two pairs can be made adjacent in the next qubit order. This can, however, also be done in the case of the linear array. □

6.4. Literature review

In this section we review the literature that has already examined the NNC problem on grid architectures. Most of the research is focused on two-dimensional grid architectures.

The authors of [54] consider a two-dimensional grid and solve the problem with an exact method for small instances, using Pseudo-Boolean Optimization. The size of the model they use experiences factorial scaling in the number of qubits for both the number of variables and the number of constraints. Their benchmarks are taken from the online library Revlib [80].

A Harmony-search heuristic algorithm is used in [2]. Harmony-search is a meta heuristic that balances factors such as randomness, experience and variation in the search. Their heuristic can quickly evaluate huge circuits of, for example, 16 qubits and almost 19.000 quantum gates. Their algorithm finds better results than the iterative look-ahead method proposed in [83], and similar results to the joint look-ahead method from the same paper.

The authors of [7] take a novel approach. Their heuristic method proposed consists of three stages. In the first stage they gather and summarize information about each qubit's interactions, in the second stage they place the qubits on a two-dimensional grid, and in the third stage, SWAP gates are inserted where needed. The method has similar results to the two-stage priority stack method of [72], where first an initial placement of the qubits is determined using the number of interactions to determine the order in which qubits are placed, and second, the required SWAP gates are inserted. Both of these methods outperform the global reordering take on the problem, which is presented in [70]. In their work, an initial placement of the qubits is optimized, and necessary deviations from the initial placement are covered by inserting SWAP gates, after which more SWAP gates are inserted to return to the initial placement.

The total number of quantum gates in a circuit is minimized in [22]. They map the problem to a boolean satisfiability problem and use a solver that performs an exhaustive search. The method scales poorly for larger circuits. They circumvent the huge search space that a large circuit would bring, by optimizing sub-circuits. Doing so, their algorithm outperforms the harmony-search method discussed in [2] on circuits of up to ten qubits and 76 quantum gates.

The authors of [33] utilize the fact that under specific commutation conditions, the gate sequence can be changed. They explore different gate orders and divide the circuit into sub-circuits such that SWAP gates are not required inside the sub-circuit. Then they use the A^* algorithm to connect the qubit orders from consecutive sub-circuits with as few SWAP gates as possible. Due to their unorthodox benchmark instances, the method has only been compared to the PAQCS method [52], which it outperforms easily.

A look-ahead heuristic is presented in [49]. Here, weights are assigned to gates, the earlier in the sequence a gate is, the higher its weight. Then, a global reordering technique is used to find a good initial placement of the qubits. To accommodate for deviations in the initial placements, a random sequence of SWAP gates is chosen such that a specific qubit order is attained. The main advantage of the method is its running time, as it can handle circuits of up to 97 qubits and 300 quantum gates, or ten qubits and just over 130.000 quantum

gates. This method, similar to the PAQCS method and the method of [33], get outperformed by the method proposed in [27]. There, the authors combine a spectral clustering algorithm and graph theory, both suitable for very large scale networks (in this case of up to 39 qubits and almost 170.000 quantum gates or 161 qubits and 50.000 quantum gates), to construct a three-stage algorithm. In the first stage, a priority queue is constructed by using the entries of the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix of the interaction graph. In the second stage, qubits are placed on a three-dimensional grid considering both the qubits that are already placed and those that still need to be placed. In the third stage, a look-ahead method is used to determine a routing procedure, inserting SWAP gates where necessary. This final method seems to be the most effective heuristic in literature to date.

7

ILP Formulations on the Grid

Now we have defined of the NNC problem in the context where qubits are placed on a two-dimensional or three-dimensional grid. The main advantage of this viewpoint is that in grid architectures, the qubits have more connections to neighboring qubits. As we saw in the previous chapter, the optimal objective value in the case of a grid is always at least as good as the optimal objective value when considering the linear array as qubit architecture. Since the main goal of this work is to minimize the overhead, we analyze the grid architectures in this chapter. Even though the objective value is at least as good as in the case of the linear array, it might be much harder to get to the optimal solution when considering two- and three-dimensional grids. To solve the nearest neighbor compliance problem where qubits are placed on a grid instead of a linear array, we return to the technique of integer linear programming.

7.1. Two-dimensional grid model

In the case where qubits are placed on a linear array, we were able to count the number of required SWAP gates when adjusting a qubit order implicitly, through the use of big-M constraints. There, we implicitly calculated the Kendall tau distance for permutations. In this case, however, where the qubits are placed on a two-dimensional grid, such a simple way of counting the number of required SWAP gates is not available.

We will still subdivide the circuit into time steps. Note that the total number of SWAP gates that acts on any fixed qubit between two gates is at most $n-1$, since a pair of qubits never have to be swapped a second time before the next gate is applied (in that case, both swaps could be left out to get the same result). This means that the total number of SWAP gates that are required to convert any qubit order into any other qubit order is at most $\gamma := \frac{n^2-n}{2}$, avoiding double counts. We will therefore introduce γ time steps in between every two consecutive gates. In each time step, only one pair of adjacent qubits is allowed to SWAP once. As before, we let $[x] := (1,\dots,x)$

7.1.1. Decision variables

First, we consider the variables. In order to keep track of the location of each qubit, binary variables x are introduced.

$$x_{iuv}^{t} = \begin{cases} 1 & \text{if qubit } i \text{ is at position } (u,v) \text{ at time } t \\ 0 & \text{else.} \end{cases} \tag{7.1}$$

We also introduce binary variables U and V in order to know whether a qubit has swapped position since the last gate was applied. Since there are now two directions qubits can be swapped in, two variables are needed to keep track of the changes. For SWAP gates between qubits that are neighbors vertically we introduce variables U,

$$U_{i}^{t} = \begin{cases} 1 & \text{if qubit } i \text{ is swapped vertically at time } t \\ 0 & \text{else,} \end{cases} \tag{7.2}$$

while for SWAP gates between horizontal neighboring qubits the variables V are used

$$V_i^t = \begin{cases} 1 & \text{if qubit } i \text{ is swapped horizontally at time } t \\ 0 & \text{else.} \end{cases} \tag{7.3}$$

In the next section we will introduce the constraints that these variables have to satisfy.

7.1.2. Constraints

In this section the constraints that are required for the ILP formulation are presented and explained. We will come across permutation constraints, SWAP counting constraints and gate adjacency constraints.

First up are the permutation constraints. Since every qubit has to be at a location at every time step, we introduce the following constraints.

$$\sum_{(u,v)\in L} x_{iuv}^t = 1 \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.4}$$

There can at most be one qubit at a location at a time. We, however, introduce dummy qubits on which no gate operations are applied to fill up all of the locations. The binary variables x must abide

$$\sum_{i\in Q^*} x_{iuv}^t = 1 \quad \forall (u,v) \in L, t \in [\gamma m]. \tag{7.5}$$

The qubits are restricted in their movement through the grid in such a way that only two of the qubits can interchange positions once with an adjacent qubit (each other).

To keep track of vertical movement, changes in x are related to the variables U

$$\sum_{(u,v)\in L} u\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \leq U_i^t \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.6}$$

$$\sum_{(u,v)\in L} u\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \geq -U_i^t \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.7}$$

and for horizontal movement the variables V are used.

$$\sum_{(u,v)\in L} v\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \leq V_i^t \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.8}$$

$$\sum_{(u,v)\in L} v\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \geq -V_i^t \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.9}$$

There is a problem with allowing all of the combinations of these movements. This is demonstrated in the following example.

Example 7.1. Let a grid of size 2×2 be given, with four qubits. If we start with any qubit order and want to rotate every qubit one position clockwise, we need three SWAP gates to do so (SWAP one qubit in counter-clockwise direction three times). If we, however, count SWAP gates as is allowed by Equation 7.26, we can simply “move" all qubits clockwise one time. This results in 4 of the U and V variables to become 1, which corresponds with a cost of only 2.

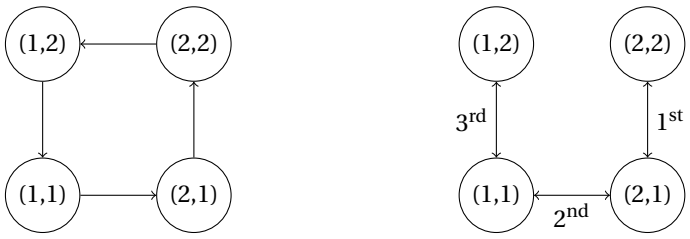


Figure 7.1: On the right: the phenomenon described in the example. The qubits all change position by moving at the same time. They obey the permutation constraints as well as the movement constraints. The situation is however not allowed since moving qubits requires the use of a SWAP gate, which does not operate in this way. On the left: the SWAP gates that are required to effectively rotate all qubits one spot. The sequence of the SWAPs is indicated by the edge labels.

To exclude the phenomenon in the example from occurring, we introduce the limitation

$$\sum_{i \in Q^*} U_i^t + V_i^t \leq 2 \quad \forall t \in [\gamma m], \tag{7.10}$$

which makes sure that the operations that we count using the U and V variables are actually SWAP gates.

The gate adjacency constraints are up next. They limit the possible qubit orders when a gate is applied such that the gate always acts on qubits that are adjacent. Note that a qubit now has four neighbors (if it is not on the edge).

$$x_{i(u-1)v}^{\gamma t} + x_{i(u+1)v}^{\gamma t} + x_{iu(v-1)}^{\gamma t} + x_{iu(v+1)}^{\gamma t} \geq x_{juv}^{\gamma t} \quad \forall g_{ij}^t \in G, (u,v) \in L \tag{7.11}$$

Terms corresponding to non-existent locations can simply be omitted form the constraint.

7.1.3. Objective function

The objective function is defined such that it counts every time when two qubits are swapped. Minimizing the objective function therefore minimizes the movement of the qubits, in particular the number of SWAP gates that are used throughout the circuit. Every SWAP gate results in the movement of two qubits, therefore only half of the changes in position are counted in the objective function. The goal is formally

$$\min \sum_{i \in Q^*} \sum_{t \in [\gamma m]} U_i^t + V_i^t \tag{7.12}$$

Notice that this objective function is closely related to the objective function of the one dimensional case. The differences are simply the increase in the number of time steps and counting the movement in the extra dimension.

7.1.4. ILP formulation

Collecting the variables, constraints and objective function into one ILP, we obtain the following program:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in Q^*} \sum_{t \in [\gamma m]} U_i^t + V_i^t & \\
\text{subject to} \quad & \sum_{(u,v) \in L} x_{iuv}^t = 1 & \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{i \in Q^*} x_{iuv}^t = 1 & \forall (u,v) \in L, t \in [\gamma m] \\
& \sum_{(u,l) \in L} u\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \leq U_i^t & \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,l) \in L} u\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \geq -U_i^t & \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,l) \in L} v\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \leq V_i^t & \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,l) \in L} v\left(x_{iuv}^t - x_{iuv}^{t+1}\right) \geq -V_i^t & \forall i \in Q^*, t \in [n^2 m] \\
& \sum_{i \in Q^*} U_i^t + V_i^t \leq 2 & \forall t \in [\gamma m] \\
& x_{i(u-1)v}^{\gamma t} + x_{i(u+1)v}^{\gamma t} + x_{iu(v-1)}^{\gamma t} + x_{iu(v+1)}^{\gamma t} \geq x_{juv}^{\gamma t} & \forall g_{ij}^t \in G, (u,v) \in L \\
& x_{iuv}^t \in \{0,1\} & \forall i \in Q, (u,v) \in L, t \in [\gamma m] \\
& U_i^t \in \{0,1\} & \forall i \in Q, t \in [\gamma m] \\
& V_i^t \in \{0,1\} & \forall i \in Q, t \in [\gamma m].
\end{aligned} \tag{7.13}$$

7.1.5. Model size

The ILP model contains substantially more variables and constraints than was the case for the linear array qubit architecture. The number of variables is

$$\#\text{variables} = \frac{n^4 - n^3}{2} m + 2n^3 m. \tag{7.14}$$

The number of constraints is

$$\#\text{constraints} = 3m(n^3 - n^2) + nm. \tag{7.15}$$

Clearly, the size of the model grows faster than was the case for the ILP model of Chapter 5, where the number of variables and the number of constraints only scaled as $\mathcal{O}(n^2 m)$.

7.2. Three-dimensional grid model

Now that the ILP model for the two-dimensional grid has been established, we turn our focus to the case in which qubits are placed on a three-dimensional grid. In a three-dimensional grid, the connectivity of the qubits is even more promising than is the case in the two-dimensional grid. The model in the previous section extends naturally to encompass a higher dimension.

As usual, we model the problem for a quantum circuit with n qubits and m quantum gates. The locations, now described by three variables, are denoted by coordinates $(u, v, w) \in L$. Similar to the previous chapter, we use γm time steps to model the problem, where only one SWAP gate is allowed per time step.

7.2.1. Decision variables

The variables in this model will, once more, all be binary. We start out with declaring variables x^t_{iuvw} that keep track of a qubit's location (u, v, w).

$$x^t_{iuvw} = \begin{cases} 1 & \text{if qubit } i \text{ is at location } (u, v, w) \text{ at time } t \\ 0 & \text{else.} \end{cases} \tag{7.16}$$

In order to keep track of qubits' movement in all three dimensions, we introduce binary variables for each qubit and each time step. Counting qubit location changes in the first dimension is identified by making use of the variables U^t_i.

$$U^t_i = \begin{cases} 1 & \text{if qubit } i \text{ is swapped with a qubit that has a different first coordinate at time } t \\ 0 & \text{else.} \end{cases} \tag{7.17}$$

In order to identify a change in a qubit's location related to movement in the second dimension, the variables V^t_i are used:

$$V^t_i = \begin{cases} 1 & \text{if qubit } i \text{ is swapped with a qubit that has a different second coordinate at time } t \\ 0 & \text{else.} \end{cases} \tag{7.18}$$

Analogous to the U^t_i and V^t_i variables, a change in a qubit's location related to the third dimension is captured with variables W^t_i:

$$W^t_i = \begin{cases} 1 & \text{if qubit } i \text{ is swapped with a qubit that has a different third coordinate at time } t \\ 0 & \text{else.} \end{cases} \tag{7.19}$$

We next introduce the constraints that are required to enforce the above definitions.

7.2.2. Constraints

In this section the constraints that are required for the ILP formulation are presented and explained. We will come across permutation constraints, SWAP counting constraints and gate adjacency constraints.

First up are the permutation constraints. Since every qubit has to be at a location at every time step, we introduce the following constraints.

$$\sum_{(u,v,w)\in L} x^t_{iuvw} = 1 \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.20}$$

There can at most be one qubit at a location at a time. Following the example of the model for the two-dimensional grid, we introduce dummy qubits on which no gate operations are applied to fill up all of the locations. The binary variables x must abide

$$\sum_{i \in Q^*} x^t_{iuvw} = 1 \quad \forall (u,v,w) \in L, t \in [\gamma m]. \tag{7.21}$$

The qubits are restricted in their movement through the grid in such a way that only two of the qubits can interchange positions once with an adjacent qubit (each other).

To keep track of movement in the first dimension, changes in x are related to the variables U,

$$\sum_{(u,v,w) \in L} u\left(x^t_{iuvw} - x^{t+1}_{iuvw}\right) \leq U^t_i \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.22}$$

$$\sum_{(u,v,w) \in L} u\left(x^t_{iuvw} - x^{t+1}_{iuvw}\right) \geq -U^t_i \quad \forall i \in Q^*, t \in [\gamma m]. \tag{7.23}$$

For movement in the second dimension, the variables V are used,

$$\sum_{(u,v,w) \in L} v\left(x^t_{iuvw} - x^{t+1}_{iuvw}\right) \leq V^t_i \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.24}$$

$$\sum_{(u,v,w) \in L} v\left(x^t_{iuvw} - x^{t+1}_{iuvw}\right) \geq -V^t_i \quad \forall i \in Q^*, t \in [\gamma m]. \tag{7.25}$$

Finally, for movement in the third dimension, the variables V are used,

$$\sum_{(u,v,w) \in L} w\left(x^t_{iuvw} - x^{t+1}_{iuvw}\right) \leq W^t_i \quad \forall i \in Q^*, t \in [\gamma m] \tag{7.26}$$

$$\sum_{(u,v,w) \in L} w\left(x^t_{iuvw} - x^{t+1}_{iuvw}\right) \geq -W^t_i \quad \forall i \in Q^*, t \in [\gamma m]. \tag{7.27}$$

We limit the movement of the qubits in such a way that only two qubits can move in such a way that they interchange their respective positions.

$$\sum_{i \in Q^*} U^t_i + V^t_i + W^t_i \leq 2 \quad \forall t \in [\gamma m], \tag{7.28}$$

which makes sure that the operations that we count using the U, V and W variables are actually SWAP gates.

The gate adjacency constraints are up next. They limit the possible qubit orders when a gate is applied such that the gate always acts on qubits that are adjacent to each other. Note that a qubit now has six neighbors (if it is not on an edge or in a corner of the grid).

$$x^{\gamma t}_{i(u-1)vw} + x^{\gamma t}_{i(u+1)vw} + x^{\gamma t}_{iu(v-1)w} + x^{\gamma t}_{iu(v+1)w} + x^{\gamma t}_{iuv(w-1} + x^{\gamma t}_{iuv(w+1)} \geq x^{\gamma t}_{juvw} \quad \forall g^t_{ij} \in G, (u,v,w) \in L \tag{7.29}$$

If a subscript refers to a non-existent location, the corresponding variable is left out of the constraint or, equivalently, set to zero.

7.2.3. Objective function

The objective is to minimize the movement of the qubits, in particular the number of SWAP gates that are used throughout the circuit. Every SWAP gate consists of the movement of two qubits, therefore only half of the changes in position are counted in the objective function. The goal is formally

$$\min \sum_{i \in Q^*} \sum_{t \in [\gamma m]} U^t_i + V^t_i + W^t_i \tag{7.30}$$

7.2.4. ILP formulation

Collecting the variables, constraints and objective function into one ILP, we obtain a complete ILP model for the NNC problem on a two-dimensional grid:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i\in Q^*} \sum_{t\in[\gamma m]} U_i^t + V_i^t + W_i^t \\
\text{subject to} \quad & \sum_{(u,v,w)\in L} x_{iuvw}^t = 1 && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{i\in Q^*} x_{iuvw}^t = 1 && \forall (u,v,w) \in L, t \in [\gamma m] \\
& \sum_{(u,v,w)\in L} u\left(x_{iuvw}^t - x_{iuvw}^{t+1}\right) \leq U_i^t && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,v,w)\in L} u\left(x_{iuvw}^t - x_{iuvw}^{t+1}\right) \geq -U_i^t && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,v,w)\in L} v\left(x_{iuvw}^t - x_{iuvw}^{t+1}\right) \leq V_i^t && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,v,w)\in L} v\left(x_{iuvw}^t - x_{iuvw}^{t+1}\right) \geq -V_i^t && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,v,w)\in L} w\left(x_{iuvw}^t - x_{iuvw}^{t+1}\right) \leq W_i^t && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{(u,v,w)\in L} w\left(x_{iuvw}^t - x_{iuvw}^{t+1}\right) \geq -W_i^t && \forall i \in Q^*, t \in [\gamma m] \\
& \sum_{i\in Q^*} U_i^t + V_i^t + W_i^t \leq 2 && \forall t \in [\gamma m] \\
& x_{i(u-1)vw}^{\gamma t} + x_{i(u+1)vw}^{\gamma t} + x_{iu(v-1)w}^{\gamma t} + x_{iu(v+1)w}^{\gamma t} + x_{iuv(w-1)}^{\gamma t} + x_{iuv(w+1)}^{\gamma t} \geq x_{juvw}^{\gamma t} && \forall g_{ij}^t \in G, (u,v,w) \in L \\
& x_{iuvw}^t \in \{0,1\} && \forall i \in Q^*, (u,v,w) \in L, t \in [\gamma m] \\
& U_i^t \in \{0,1\} && \forall i \in Q^*, t \in [\gamma m] \\
& V_i^t \in \{0,1\} && \forall i \in Q^*, t \in [\gamma m] \\
& W_i^t \in \{0,1\} && \forall i \in Q^*, t \in [\gamma m]. \qquad (7.31)
\end{aligned}
$$

7.2.5. Model size

The size of this model scales once again polynomially in the number of qubits and quantum gates. The number of variables in this model is

$$\#\text{variables} = \frac{n^4 - n^3}{2} m + 3n^3 m. \tag{7.32}$$

The number of constraints in the model is

$$\#\text{constraints} = 4n^3 m - 2n^2 m + nm. \tag{7.33}$$

Notice that the smallest three-dimensional grid is of size $2 \times 2 \times 2$. This can already accommodate for eight qubits. In practice, this will result in too many variables and constraints already.

8

Experimental Results and Discussion

The integer linear programming models that were introduced in Chapters 5 and 7, model the NNC problem in the cases of a linear array, the two-dimensional grid, and the three-dimensional grid. The former two of those models are used to find exact solutions to 131 benchmark instances. The size of the formulation that models the NNC problem on a three-dimensional grid scales too rapidly to evaluate any benchmark instances. Note that a quantum circuit can only make use of the third dimension if we consider eight or more qubits.

In this chapter, the exact solutions provided by the proposed models are compared to the previous best exact approaches in terms of instance size they can cope with, as well as state-of-the-art heuristic approaches in terms of attained objective value.

The mathematical models, as described in previous chapters, have been implemented in Python and solved with the commercial solver CPLEX 12.7 through the Python API[1]. All but the Quantum Fourier Transform (QFT) instances[2], which were constructed following the circuit of [63], were obtained form the RevLib [80] website. The evaluations were conducted using up to 16 threads of 2.4 GHz each, working with 16 GB of RAM in total. Optimal solutions were acquired for all of the evaluated instances.

8.1. Results: Linear array

The benchmark instances are divided over four tables, according to the number of qubits in the addressed quantum circuit. In the first column of each table, the name of the circuit is provided, in the second column, n denotes the number of qubits in the circuit. In the third column, $|G|$ denotes the number of two-qubit gates present in the circuit after gate decomposition and after the removal of single-qubit gates. The optimal value of the local reordering problem, i.e., the minimum number of SWAP gates needed to make the circuit nearest neighbor compliant, is provided in the fourth column. The column "Time" denotes the running time of our approach, in seconds. The column titled "Time E" denotes the running time of other exact methods, also in seconds. Exact running times with subscript a are from [82], subscript b from [57]. The objective value of heuristically determined solutions are presented in the last column, denoted by "# SWAPS H". Here, the subscript c indicates the results are from [49], subscript d from [69], subscript e from [1], subscript f from [48], subscript g from [83], and subscript h from [7]. An asterisk as superscript indicates that for the other exact solution methods, either the objective value differs, or the number of gates differs or they both differ. For the heuristic results, the asterisk indicates that the number of gates differs or the objective value of the heuristic is lower than that of the proposed exact method. These anomalies are believed to have their roots in differing gate decomposition methods, resulting in slightly different instances.

The running time that is required to solve an instance depends heavily on three factors:

1. The number of qubits in the quantum circuit,
2. The number of gates in the quantum circuit,

[1] See: https://www.ibm.com/analytics/cplex-optimizer

[2] The nearest neighbor compliant circuit that resulted from evaluating the QFT circuit on five qubits, is showcased on the frontpage of this work.

3. The minimal number of required SWAP gates.

The number of qubits and gates is expected to influence the running time heavily. The number of qubits is the term that influences the run time the most. This is due to the fact that the number of feasible solutions scales factorially in the number of qubits. Surprisingly, the run time also scales quite badly with the number of required SWAP gates. During the Branch & Bound tree search, the upper bound determined by CPLEX, which is the best feasible solution found up to that point, converges to the optimal value (or close to it) rather quickly. The best known lower bound, however, takes a long time to improve. When the number of required SWAP gates increases, the time that is needed to improve the lower bound all the way to the optimal value increases as well. This phenomenon is analyzed for two of the benchmark instances that require a lot of SWAP gates:

1. **mod8-10_177** The search method found a feasible solution with an objective value within 10% of the optimal value in $2.4 \cdot 10^6$ iterations, found an optimal solution in $4.0 \cdot 10^7$ iterations, and proved optimality by finding a matching lower bound after $1.7 \cdot 10^8$ iterations. Completing the search took 3650 seconds.

2. **decod24-enable_126** The search method found a feasible solution with an objective value within 10% of the optimal value in $5.3 \cdot 10^6$ iterations, found an optimal solution in $1.0 \cdot 10^7$ iterations, and proved optimality by finding a matching lower bound after $5.8 \cdot 10^7$ iterations. Completing the search took 1954 seconds.

If a 10% optimality gap would suffice, only less than 2% of the total number of iterations would be needed in the first case, and 10% in the second case. This observation indicates that running an incomplete Branch & Bound algorithm might be an interesting and easy-to-implement heuristic algorithm.

The 131 evaluated benchmark instances are listed in the Tables 8.1, 8.2, 8.3 and 8.4. The improvement in computation time with respect to previous exact methods is significant. The results show exact solutions that are obtained for much larger circuits than previously held possible. The largest instance with respect to the number of qubits has as much as 18 qubits. Furthermore, for the first time, NNC has been solved to optimality for circuits with more than 100 quantum gates.

Table 8.1: Benchmark instances with three or four qubits, evaluated on a linear array.

Benchmark	n	$\|G\|$	# SWAPS	Time	Time E	# SWAPS H
QFT_QFT3	3	3	1	0.02	-	-
peres_10	3	4	1	0.14	0.1_a	-
peres_8	3	4	1	0.06	0.1_a	-
toffoli_2	3	5	1	0.12	0.2_a	-
toffoli_1	3	5	1	0.1	0.1_a	-
peres_9	3	6	1	0.02	2463_a	-
fredkin_7	3	7	1	0.16	-	-
ex-1_166	3	7	2	0.08	0.1_a	-
fredkin_5	3	7	1	0.15	$0.1_a, 0.1^*_b$	-
ham3_103	3	8	2	0.04	-	-
miller_12	3	8	2	0.14	$745.6_a, 0.1_b$	-
ham3_102	3	9	1	0.05	0.1^*_a	-
3_17_15	3	9	2	0.04	$630.2_a, 0.1^*_b$	-
3_17_13	3	13	3	0.12	0.1^*_a	$4^*_c, 4_d, 3_e, 6_g, 6_h$
3_17_14	3	13	3	0.15	0.1^*_a	-
fredkin_6	3	15	3	0.06	4.6_a	-
miller_11	3	17	4	0.15	0.1^*_a	-
QFT_QFT4	4	6	3	0.17	-	-
toffoli_double_3	4	7	1	0.11	$0.9_a, 0.1^*_b$	-
rd32-v1_69	4	8	2	0.16	0.1_a	-
decod24-v1_42	4	8	2	0.12	$7.7_a, 0.1^*_b$	-
rd32-v0_67	4	8	2	0.07	1.6_a	$2_c, 2_d, 4_h$
decod24-v2_44	4	8	3	0.07	0.1^*_b	-
decod24-v0_40	4	8	3	0.06	0.1^*_b	-
decod24-v3_46	4	9	3	0.09	$0.1_a, 0.1^*_b$	$3_c, 3_d$
toffoli_double_4	4	10	2	0.07	200^2_a	-
rd32-v1_68	4	12	3	0.24	0.4^*_a	-
rd32-v0_66	4	12	0	0.09	0.4^*_a	-
decod24-v0_39	4	15	5	0.53	0.5_a	-
decod24-v2_43	4	16	5	0.23	0.1^*_a	-
decod24-v0_38	4	17	4	0.57	19.2_a	-
decod24-v1_41	4	21	7	0.5	-	-
hwb4_52	4	23	8	0.97	-	$9_c, 10_d, 9_e, 9_f, 9_h$
aj-e11_168	4	29	12	5.36	-	-
4_49_17	4	30	12	6.1	-	$12^*_c, 12_d, 16_e, 15^*_h$
decod24-v3_45	4	32	13	6.25	-	-
mod10_176	4	42	15	7.94	-	-
aj-e11_165	4	44	18	9.36	-	$36_d, 33^*_g, 33^*_h$
mod10_171	4	57	24	27.18	-	-
4_49_16	4	59	22	24.23	-	-
mini-alu_167	4	62	27	23.7	-	-
hwb4_50	4	63	23	17.61	-	-
hwb4_49	4	65	23	21.64	-	-
hwb4_51	4	75	28	75.09	-	-

Table 8.2: Benchmarks with five qubits, evaluated on a linear array.

Benchmark	n	$\|G\|$	# SWAPS	Time	Time E	# SWAPS H
4mod5-v1_25	5	7	1	0.26	11705.3_a	-
4gt11_84	5	7	1	0.06	16.6_a	$1_c, 1_d, 1_e, 4_h$
4gt11-v1_85	5	7	1	0.09	-	-
4mod5-v0_20	5	8	2	0.08	45.5_a	-
4mod5-v1_22	5	9	1	0.08	548.8^*_a	-
QFT_QFT5	5	10	6	0.41	1.6_a	$7_c, 6_d$
mod5d1_63	5	11	2	0.12	-	-
4mod5-v0_19	5	12	3	0.84	55.3^*_a	-
4gt11_83	5	12	3	0.15	9^*_a	-
4mod5-v1_24	5	12	3	0.28	-	-
mod5mils_65	5	12	4	0.26	-	-
mod5mils_71	5	12	2	0.15	-	-
alu-v2_33	5	13	4	0.45	-	-
alu-v1_29	5	13	4	0.61	-	-
alu-v0_27	5	13	4	0.48	-	-
mod5d2_70	5	14	5	0.43	-	-
alu-v3_35	5	14	5	0.38	-	-
alu-v4_37	5	14	5	0.37	-	-
alu-v1_28	5	14	4	0.26	-	-
4gt13-v1_93	5	15	5	0.69	489.3^*_a	$7^*_c, 6_d, 4^*_e, 9^*_h$
4gt13_92	5	15	6	0.53	-	-
4gt11_82	5	16	6	0.89	-	-
4mod5-v0_21	5	17	8	2.84	-	-
rd32_272	5	18	7	0.94	-	-
alu-v3_34	5	18	4	0.4	-	-
mod5d2_64	5	19	6	1.81	-	-

Table 8.3: Benchmarks with five qubits, evaluated on a linear array.

Benchmark	n	$\|G\|$	# SWAPS	Time	Time E	# SWAPS H
alu-v0_26	5	21	8	3.56	-	-
4gt5_75	5	21	6	1.1	-	$9^*_c, 12_d, 13^*_h$
4mod5-v0_18	5	23	8	3.35	-	-
4mod5-v1_23	5	24	9	5.06	-	$9_c, 9_d, 15_e, 15_h$
one-two-three-v2_100	5	24	7	5.37	-	-
one-two-three-v3_101	5	24	7	2.96	-	-
rd32_271	5	26	11	7.37	-	-
4gt5_77	5	28	10	6.2	-	-
4gt5_76	5	29	10	5.45	-	-
alu-v4_36	5	30	9	6.34	-	$15^*_c, 18_d, 17_e, 16^*_h$
4gt13_91	5	30	8	4.46	-	-
4gt13_90	5	34	12	6.77	-	-
4gt10-v1_81	5	34	13	12.38	-	$18^*_c, 20_d, 16_e, 24^*_g, 22^*_h$
one-two-three-v1_99	5	36	15	17.27	-	-
4gt4-v0_80	5	36	19	43.45	-	$34_d, 33_f, 32^*_h$
4mod7-v0_94	5	38	12	12.83	-	-
alu-v2_32	5	38	16	22.05	-	-
4mod7-v0_95	5	38	14	14.59	-	$19^*_c, 21_d, 22_e, 22^*_h$
4mod7-v1_96	5	38	14	13.49	-	-
one-two-three-v0_98	5	40	15	15.67	-	-
4gt12-v0_88	5	41	20	34.01	-	-
4gt12-v1_89	5	44	22	52.36	-	$35_d, 26_e, 32_f, 33^*_h$
sf_275	5	46	18	21.42	-	-
4gt4-v0_79	5	49	22	80.16	-	-
4gt4-v0_78	5	53	26	167.03	-	-
4gt4-v0_72	5	53	24	49.7	-	-
4gt12-v0_87	5	54	22	45.88	-	-
4gt4-v1_74	5	57	29	84.87	-	-
4gt12-v0_86	5	58	26	108.35	-	-
mod8-10_178	5	68	37	389.47	-	-
one-two-three-v0_97	5	71	32	76.8	-	-
4gt4-v0_73	5	89	40	699.65	-	-
mod8-10_177	5	93	48	3650.26	-	$72_d, 71^*_h$
alu-v2_31	5	100	49	2906.35	-	-
hwb5_55	5	101	48	2264.0	-	$59_c, 63_d, 60_e, 66_g, 64^*_h$
rd32_273	5	104	50	4631.7	-	-
alu-v2_30	5	112	55	13558.87		

Table 8.4: Benchmarks with six or more qubits, evaluated on a linear array.

Benchmark	n	$\|G\|$	# SWAPS	Time	Time E	# SWAPS H
graycode6_47	6	5	0	0.02	-	-
graycode6_48	6	5	0	0.02	-	-
QFT_QFT6	6	15	11	7.43	-	$11_c, 12_d$
decod24-enable_124	6	21	5	1.86	-	-
decod24-enable_125	6	21	5	1.83	-	-
decod24-bdd_294	6	24	7	9.37	-	-
mod5adder_129	6	71	34	534.38	-	-
mod5adder_128	6	77	36	1103.51	-	$45^*_c, 51_d, 46^*_g, 53^*_h$
decod24-enable_126	6	86	37	1954.28	-	-
xor5_254	7	5	3	0.61	-	-
ex1_226	7	5	3	0.25	-	-
QFT_QFT7	7	21	16	28.26	-	$28_c, 26_d, 18_g, 18_h$
4mod5-bdd_287	7	23	7	4.3	-	-
alu-bdd_288	7	28	8	20.65	-	-
ham7_106	7	49	28	495.43	-	-
ham7_105	7	65	34	1613.33	-	-
ham7_104	7	83	42	3238.82	-	$56^*_c, 66^*_h$
QFT_QFT8	8	28	23	334.6	-	$32_c, 33_d, 31_g, 31_h$
rd53_139	8	36	11	76.29	-	-
rd53_138	8	44	11	100.86	-	-
rd53_137	8	66	35	6271.11	-	-
QFT_QFT9	9	36	30	1482.53	-	$52_c, 54_d, 49_g, 42_h$
QFT_QFT10	10	45	39	39594.99	-	$64_g, 60_h$
mini_alu_305	10	57	23	1711.75	-	-
sys6-v0_144	10	62	19	887.71	-	-
rd73_141	10	64	21	845.05	-	-
parity_247	18	16	14	5762.29	-	-

8.2. Results: Two-dimensional grid

In this section, a subset of the benchmark instances that were evaluated in Section 8.1, are evaluated for the two-dimensional grid architecture. The benchmark instances are divided into three categories, dependent on the number of qubits in the quantum circuit. The grid that was used, is presented in the final column of the tables. The largest instances in the two-dimensional case are substantially smaller than the largest instances in the case where qubits are placed on a linear array. By Theorem 6.4, the optimal objective value that is obtained in the two-dimensional case, is always bounded from above by the optimal objective value that was obtained in the case of the linear array. Theorem 6.8 states that the results in Table 8.5 have the same optimal objective value as the results corresponding to circuits with three qubits in Table **??**.

Table 8.5: Benchmark instances with three qubits, evaluated on a two-dimensional grid.

Benchmark	n	$\|G\|$	# SWAPS	Time	Grid
QFT_QFT3	3	3	1	0.16	2×2
peres_10	3	4	1	0.25	2×2
peres_8	3	4	1	0.23	2×2
toffoli_2	3	5	1	0.29	2×2
toffoli_1	3	5	1	0.28	2×2
peres_9	3	6	1	0.34	2×2
fredkin_7	3	7	1	0.64	2×2
ex-1_166	3	7	2	0.64	2×2
fredkin_5	3	7	1	0.51	2×2
ham3_103	3	8	2	0.8	2×2
miller_12	3	8	2	0.7	2×2
ham3_102	3	9	1	0.89	2×2
3_17_15	3	9	2	0.91	2×2
3_17_13	3	13	3	3.8	2×2
3_17_14	3	13	3	4.18	2×2
fredkin_6	3	15	3	5.26	2×2
miller_11	3	17	4	9.04	2×2

In Table 8.5 we see that solving the problem for benchmark instances that have three qubits, using the model for the two-dimensional architecture, is done in under ten seconds as long as the gate count is not too large. We see, in Table 8.6, that the running time already approaches practical limits for the benchmark instances with the most gates. The optimal objective value in the two-dimensional case is much better than the one in the one-dimensional case of the NNC problem. For example, the circuit “4_49_17” only requires six SWAP gates when implemented on a two-dimensional grid while the same circuit requires twelve SWAP gates when implemented on a linear array.

The circuits that also benefit from being implemented on a higher-dimensional grid, are the famous Quantum Fourier Transform (QFT) circuits. The QFT for five qubits only requires four SWAP gates when implemented on a two-dimensional grid, where we require six SWAP gates when implementing it on a linear array.

8.3. Discussion

The fourth research question: "What are the main advantages and disadvantages of exact solution methods for the nearest neighbor compliance problem?" is discussed in this section. We explore advantages and disadvantages of our methods and argue why certain choices have been made in this work.

The exact solution methods that we use, also have drawbacks. A major disadvantage is that they can only evaluate small circuits. Even though their solutions might not be (proven to be) optimal, heuristic methods can evaluate much larger benchmark instances. As we have seen in the literature review, even on three dimensional architectures, circuits of 39 qubits and 170.000 quantum gates, or as much as 161 qubits and 50.000 quantum gates can be evaluated. The quality of the solutions is hard to predict, but a solution is obtained nonetheless. Notice that the incomplete Branch & Bound search in the previous section was used to evaluate a circuit of only five qubits and 93 quantum gates.

Table 8.6: Benchmark instances with 4 qubits and up to 30 gates, evaluated on a two-dimensional grid.

Benchmark	n	$\|G\|$	# SWAPS	Time	Grid
QFT_QFT4	4	6	2	0.46	2×2
toffoli_double_3	4	7	1	0.53	2×2
rd32-v1_69	4	8	2	1.27	2×2
decod24-v1_42	4	8	0	0.35	2×2
rd32-v0_67	4	8	2	1.13	2×2
decod24-v2_44	4	8	0	0.31	2×2
decod24-v0_40	4	8	0	0.34	2×2
decod24-v3_46	4	9	2	0.94	2×2
toffoli_double_4	4	10	2	1.06	2×2
rd32-v1_68	4	12	2	2.47	2×2
rd32-v0_66	4	12	2	2.64	2×2
decod24-v0_39	4	15	5	41.47	2×2
decod24-v2_43	4	16	5	43.28	2×2
decod24-v0_38	4	17	4	13.56	2×2
decod24-v1_41	4	21	7	133.52	2×2
hwb4_52	4	23	6	106.04	2×2
aj-e11_168	4	29	9	1126.04	2×2
4_49_17	4	30	6	182.27	2×2

Table 8.7: Benchmark instances with 5 and 6 qubits, evaluated on a two-dimensional grid.

Benchmark	n	$\|G\|$	# SWAPS	Time	Grid
4mod5-v1_25	5	7	1	55.17	2×3
4gt11_84	5	7	1	91.49	2×3
4gt11-v1_85	5	7	1	56.93	2×3
4mod5-v0_20	5	8	1	101.4	2×3
4mod5-v1_22	5	9	1	130.63	2×3
QFT_QFT5	5	10	4	4885.45	2×3
mod5d1_63	5	11	1	154.42	2×3
4gt11_83	5	12	1	256.4	2×3
4mod5-v1_24	5	12	2	2499.35	2×3
4mod5-v0_19	5	12	3	8593.68	2×3
graycode6_47	6	5	0	2.77	2×3
graycode6_48	6	5	0	2.84	2×3

Another discussion point of our exact method, is that it produces exact solutions to a problem that considers limited options. We distinguish three aspects of quantum circuit design that we disregard in our optimization methods.

First, we do not optimize over the different gate decomposition possibilities. Multi-qubit gates can be decomposed into two-qubit gates in many ways. If we were to take this option into consideration, we obtain an instance of the Minimum Generator Sequence problem. Here, the calculation done by a circuit corresponds to a group element of $\mathbb{C}^{2^n \times 2^n}$, where n is the number of qubits. The generators of the group would in that case be the unitary matrices corresponding to the operations of two-qubit gates acting on adjacent qubits. The number of feasible solutions grows rapidly as the calculation performed by a quantum circuit becomes more complex, therefore, the only circuits that can be evaluated would likely be even smaller compared to the current method.

Second, we do not take into account different gate sequences while it could sometimes be advantageous to do so in terms of how many SWAP gates are required. In some cases, the gate sequence can be changed. The circumstances under which this is possible, are governed by a set of commutation conditions. If, for example, two consecutive gates act on disjoint pairs of qubits, the gates might as well be performed in reverse

order. Taking changes in the gate sequence into account was already done in [57], but by making use of the exponentially growing adjacent transposition graph. It is not clear how the option to make changes to the gate sequence can be efficiently incorporated into the current methodology.

A third aspect of circuit design that is not properly addressed in our methodology is the depth of a circuit. As mentioned in Chapter 2, the coherence times of qubits are a limiting factor in the time a quantum circuit may take to complete running. Throughout our research, we have assumed that all gates are applied sequentially. If, for example, two gates were to act on disjoint pairs of qubits, they could be executed simultaneously in parallel, saving valuable time. The problem of Makespan Optimization (MO), which was considered by the authors of [10], minimizes the total time it takes for a circuit to complete running. The MO problem is NP-complete.

II

Related Research Areas

9

Research Area II: Global Reordering

In this chapter we will discuss the Global Reordering problem, which corresponds to research area II from Table 1.1. We do not introduce any new concepts, ideas or methods. As such, this chapter can best be thought of as a review of the currently available literature. We address the problem because it is one of the major constituents of the field of nearest neighbor quantum circuit design. The current literature is reviewed and the main results are explained.

9.1. Introduction to global reordering

The problem of *global reordering* is popular in the field of nearest neighbor quantum circuit design. Global reordering concerns itself with finding an optimal initial placement of the qubits, such that throughout the circuit, as few SWAP gates as possible are required. The problem is a simplified version of local reordering since it eludes the micromanagement of smartly swapping qubits around between gates, with which the local reordering problem is imbued.

The biggest advantage of global reordering over local reordering is its simplicity (in comparison). The size of the search space does not grow as rapidly as is the case with local reordering. The complexity does not depend on the number of gates in a quantum circuit.

In an attempt to keep an edge over local reordering in terms of simplicity, the global reordering problem is often approximated using the Optimal Linear Arrangement (OLA) problem. In this chapter we will discuss the model that is used to assess the global reordering problem via an implementation of the OLA problem.

The Linear Nearest Neighbor problem is formulated using Mixed Integer Programming in [70]. In this case, there is no hard constraint for qubits to be adjacent when a gate acts on them. The objective is instead to minimize the distance between qubits that interact a lot with each other. This problem can be seen as a global reordering technique. In general, however, this does not guarantee an optimal initial ordering for the local reordering approach. A global strategy is devised using an analogy to the Optimal Linear Arrangement Problem [69],

9.2. Optimal Linear Arrangement problem

In this section we introduce the Optimal Linear Arrangement problem. The decision problem is defined as follows:

OPTIMAL LINEAR ARRANGEMENT PROBLEM

Input: A graph $G = (V, E)$ with $|V| = n$, a function $w : E \to \mathbb{R}$ and a number $k \in \mathbb{R}$.
Question: Does there exist a bijection $f : V \to [n]$ such that $\sum_{\{u,v\}\in E} w_{uv}|f(u) - f(v)| \leq k$?

In the minimization version of the OLA problem, we look for the smallest integer k such that the question is still answered affirmatively. Notice that the bijection f directly relates to a qubit order. The qubit order can be thought of as the optimal average qubit order throughout the circuit, from which one can diverge when a specific gate is not NN compliant given the global qubit order. In the general version of the OLA problem, the weights on the edges on the graph often represent costs or travel times for example.

9.3. Mathematical Model

The Optimal Linear Arrangement problem requires a graph as input. In this section, we introduce the interaction graph and its weights. We then use the OLA problem in such a way that the number of required SWAP gates matches the objective function.

9.3.1. Interaction graph

The graph that will be used is called an interaction graph. A node is constructed for each qubit in the quantum circuit. Every two nodes are then connected with an edge. The weight $w_{u,v}$ on edge $\{u,v\}$ is then calculated in one of two ways.

The first way to calculate the weights on the edges of the interaction graph is proposed in [47] and counts the number of gates between the two qubits.

$$w_{u,v} = \#\{g_{uv} \in G\} = \sum_{t=1}^{m} \mathbb{1}_{g_{uv}^t} \tag{9.1}$$

Here $\mathbb{1}_{g_{uv}^t}$ is the indicator function, which is one when g^t acts on qubits u and v, and zero otherwise. This approach is intuitively very easy to grasp, but also naive in the sense that the sequence of the gates is completely disregarded. Qubits may be placed close together initially, solely because they have a lot of gates together near the end of the circuit. The second proposal [49] circumvents this problem. It makes the sum weighted in such a way that gates that appear later in the gate sequence have less influence on the edge weights.

$$w'_{u,v} = \sum_{t=1}^{m} \frac{\mathbb{1}_{g_{uv}^t}}{t^K} \tag{9.2}$$

for some integer $K \geq 1$.

To clarify the graph, an example is given.

Example 9.1. Suppose we are given the graph of Figure 9.1. We want to construct the corresponding interaction graph. For simplicity we stick to the edge weights that do not incorporate the gate sequence. Every qubit gets its own node in the graph and an edge is placed between every two qubits that share a quantum gate. Since qubits a and e share three gates, the weight on edge $\{a,e\}$ is $w_{ae} = 3$.

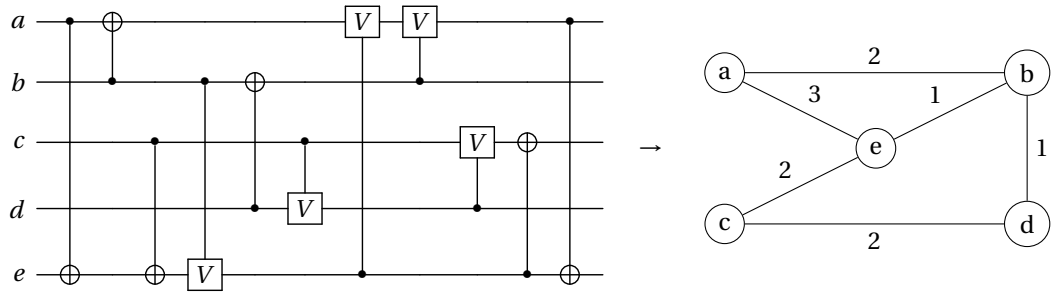


Figure 9.1: A quantum circuit with 5 qubits and 11 quantum gates is shown on the left. The corresponding interaction graph is displayed on the right. Three quantum gates are applied on the qubits a and e throughout the circuit, so the weight on edge $\{a,e\}$ is equal to $w_{ae} = 3$.

9.3.2. Nearest neighbor cost

The goal is to minimize the number of required SWAP gates in the circuit. In order to do so, we link the objective function of the OLA problem to a cost function. The cost function was introduced in [47] as the

nearest neighbor cost. Given a qubit order $\tau \in \mathscr{S}_n$ and a gate $g = \{i, j\} \in G$, the nearest neighbor cost of that gate is

$$\mathrm{NNC}(g) = 2(|\tau(i) - \tau(j)| - 1). \tag{9.3}$$

The function exactly determines how many SWAP gates are required to complete the operation: 1) change the global qubit τ_0 order to a qubit order τ_1, where qubits i and j are adjacent, 2) perform gate g, and 3) change the qubit order τ_1 back to the global qubit order τ_0. We make the observation that, given a global qubit order τ_0, adding the nearest neighbor costs of all gates together, provides an upper bound on the optimal value of the local reordering problem in the same context.

Sometimes, it might not be optimal to go back to the global qubit order immediately after applying a gate. For example, when two gates $g^t = g^{t+1} = \{i, j\}$ are applied on the same qubits consecutively, less SWAP gates are required than indicated by the nearest neighbor cost.

9.3.3. Combined model

Combining the nearest neighbor cost, the weighted interaction graph and the OLA problem, we describe the global reordering problem with the following minimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{\{u,v\} \in E} 2 w_{uv} \left(|f(u) - f(v)| - 1 \right) \\ \text{subject to} \quad & f : V \to [n] \\ & f \text{ is a bijective map.} \end{aligned} \tag{9.4}$$

One can see that this is an instance of the OLA minimization problem, by taking the constant 2 and the constant subtraction of 1 from the absolute difference outside of the minimization.

The nearest neighbor cost has the property that the number of SWAP gates required to diverge from the global qubit order to make two qubits adjacent, and change back to the global qubit order, is exactly counted for each gate. Program 9.4 sums this cost for all the gates. We therefore arrive at this intermediate result.

Theorem 9.2. *Given a quantum circuit Γ, let G be the weighted (via Equation 9.1) interaction graph corresponding to Γ. Denote the optimal value of Program 9.4 by OPT_{glo}. Denote the optimal value of the NNC problem on a linear array, considering the same quantum circuit Γ, as OPT_{1D}. Then we find the following upper bound for OPT_{1D}.*

$$OPT_{glo} \geq OPT_{1D} \tag{9.5}$$

Continuing, the global reordering problem was formulated as a Quadratic Assignment problem by the authors of [70]. They first define $c_{ijkl} = w_{ik}|f(j) - f(l)|$, where i and k represent qubits while j and l represent locations. Their model then reads

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} c_{ijkl} x_{ij} x_{kl} & \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{ij} = 1 & \forall j \in L \\ & \sum_{j=1}^{n} x_{ij} = 1 & \forall i \in Q \\ & x_{ij} \in \{0, 1\} & \forall i \in Q, j \in L \end{aligned} \tag{9.6}$$

This program was already linearized in [41] and shown to have a minimal number of variables and constraints in [12]. We define a new variable z.

$$z_{ij} = x_{ij} \sum_{k=1}^{n} \sum_{l=1}^{n} c_{ijkl} x_{kl} \tag{9.7}$$

The resulting mixed integer linear program is equivalent to Program 9.6.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} z_{ij} \\ \text{subject to} \quad & \alpha_{ij}x_{ij} + \sum_{k=1}^{n}\sum_{l=1}^{n} c_{ijkl}x_{kl} - z_{ij} \leq \alpha_{ij} \qquad \forall i \in Q, j \in L \\ & z_{ij} \leq 0 \qquad \forall i \in Q, j \in L \end{aligned} \tag{9.8}$$

Here, $\alpha_{ij} = \sum_{k=1}^{n}\sum_{l=1}^{n} c_{ijkl}$. We have arrived at a mixed integer linear programming model for the global reordering problem.

10

Research Area III: Local Reordering of Qubits in a Distributed Quantum Circuit

This Chapter was written in collaboration with Roy v. Houte.

Now we switch our focus to the problem of local reordering in the distributed case. This problem belongs to research area III, as indicated in Table 1.1. This problem is an extension of the main focus of this thesis to incorporate the option of performing a quantum circuit using multiple quantum computers. We will consider the most basic of cases, where each computer has an internal qubit architecture of a linear array, and only the qubits on the endpoints of the array are connected to the next computer. The computer network thus also forms a linear array.

In this chapter, we will consider only SWAP operations as viable communication tools between different computers, while in fact any controlled operation could be performed. This assumption is made in order to reduce the available tool set to the insertion of SWAP gates, a setting we have explored thoroughly so far. Fortunately, an extension of the framework that was introduced in Chapter 5 to this problem is natural. A paper on the results of this chapter has been submitted for possible publication [77].

10.1. Model

We start out by introducing notation. Suppose we have a quantum circuit, consisting of m unitary two-qubit gates $g_{il} \in G$, acting on a total of n qubits $\{q_1, \ldots, q_n\} \equiv Q$. The physical locations of the qubits are distributed between N quantum computers in a linear fashion, i.e., locations $L_1 = (1, \ldots, k_1)$ belong to computer C_1 and locations $L_2 = (k_1 + 2, \ldots, k_1 + k_2 + 1)$ belong to computer C_2, locations $L_N = (k_1 + \ldots + k_{N-1} + N, \ldots, k_1 + \ldots + k_N + N - 1)$ to computer C_N, where k_i is the qubit capacity of computer C_i. Here, one qubit location is skipped between every two consecutive computers, we will see that this helps with the modeling later on. Of course, we also have to comply with nearest neighbor interaction constraints, where gates can only act on two qubits if the corresponding qubits are physically adjacent, so if their locations are l_i, l_{i+1} respectively for some i.

The goal in this version of the NNC problem consists of two parts:

1 Minimize the number of SWAP gates between different computers, associated with a cost of α

2 Minimize the number of SWAP gates within each computer, associated with a cost of β

An illustration is provided for clarification in Figure 10.1.

In order to extend the ILP formulation of minimizing the number of SWAP gates in the case of one computer, as was done in Chapter 5, such that it also encapsulates the distributed variant of the local reordering problem, no big extension is required. The proposed mathematical model is presented below.

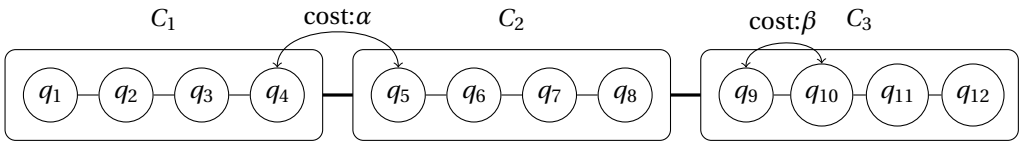


Figure 10.1: A line graph on a network of three quantum computers, indicated by the C's. The computers each have a capacity of four qubits that are also connected in a linear array.

10.1.1. Variables and constraints

Let us first introduce integer variables x_i^t, indicating the location $l \in \cup_{i\in[N]} L_i$ of a qubit q_i just before gate g^t is applied. Note that this also implies on which quantum computer the qubit is located. To count the required number of SWAP gates when changing the qubit order between gates, binary variables y_{il}^t are introduced that keep track of the pairwise ordering of two qubits q_i, q_l before gate g^t.

$$y_{il}^t = \begin{cases} 1 & \text{if } x_i^t > x_l^t \\ 0 & \text{otherwise.} \end{cases} \tag{10.1}$$

Changes in the y-variables, when moving from one gate to the next, mean that two qubits have changed their pairwise order. We, once again, implicitly use the Kendall tau metric on the permutation group to count the number of inversions and thus the number of required SWAP gates. The nearest neighbor constraints state that a gate can only act on two adjacent qubits, $|x_i^t - x_l^t| \leq 1$. This constraint is linearized using the inequalities

$$x_i^t - x_l^t \leq 1 \qquad \forall g_{il}^t \in G, \tag{10.2}$$
$$x_i^t - x_l^t \geq -1 \qquad \forall g_{il}^t \in G. \tag{10.3}$$

To keep track of the qubit order and to make sure that the distance between qubits is at least 1, the following big-M type constraints are added.

$$x_i^t - x_l^t \leq M y_{il}^t - 1 \qquad \forall i, l \in Q, i < l, t \in [m], \tag{10.4}$$
$$x_l^t - x_i^t \leq M(1 - y_{il}^t) - 1 \qquad \forall i, l \in Q, i < l, t \in [m], \tag{10.5}$$

where the constant M is chosen to be large enough, in this case $M = n + \sum_i k_i$ will suffice, to make one constraint trivially satisfied. The binary variable y determines for which of the two constraints that holds.

Recall the auxiliary locations left between the quantum computers that were supposed to help us. We will refer to these locations as "borders". These borders b are locations indexed by the values

$$b_s = s + \sum_{j\in[s]} k_j \quad s \in [N-1], \tag{10.6}$$

where b_s is the location of the border between quantum computers s and $s+1$. We want to distinguish between SWAP gates that are used to interchange qubits' locations within a computer, and SWAP gates that interchange two qubits from neighboring computers. To keep track of qubits that are swapped to a different computer, binary variables y_{is}^t are introduced.

$$y_{is}^t = \begin{cases} 1 & \text{if } x_i^t > b_s \\ 0 & \text{otherwise.} \end{cases} \tag{10.7}$$

They tell us on which side of the auxiliary location between two computers a qubit is located before gate g^t. If the qubit changes order with the auxiliary location, we add a cost to the objective function. The y_{is}^t-variables are constrained in the following way:

$$x_i^t - b_s \leq M y_{is}^t - 1 \qquad \forall i \in Q, i < l, t \in [m], s \in [N-1], \tag{10.8}$$
$$b_s - x_i^t \leq M(1 - y_{is}^t) - 1 \qquad \forall i \in Q, i < l, t \in [m], s \in [N-1]. \tag{10.9}$$

Here, the variable y_{is}^t is 0 if the location x_i^t of qubit q_i is smaller than the location of the border between computers s and $s+1$. The absolute change (from gate to gate) in the y-variables adds a cost to the objective function. Here, we again remark that the absolute value is not a linear function, so to cope with this, we linearize the function by introducing new variables p and r. The p variables are used for costs relating to SWAP gates within a computer:

$$p_{il}^t = \begin{cases} 1 & \text{if the order of qubits } i \text{ and } l \text{ changed from gate } g^t \text{ to } g^{t+1} \\ 0 & \text{otherwise.} \end{cases} \tag{10.10}$$

The r variables are related to costs regarding SWAP gates between qubits on different quantum computers:

$$r_{is}^t = \begin{cases} 1 & \text{if qubit } i \text{ crossed border } b_s \text{ between gates } g^t \text{ and } g^{t+1} \\ 0 & \text{otherwise.} \end{cases} \tag{10.11}$$

To enforce their definitions, we introduce constraints for both new types of variables. The p's are constrained as

$$y_{il}^t - y_{il}^{t+1} \leq p_{il}^t \qquad \forall i,l \in Q, i < l, t \in [m-1], \tag{10.12}$$
$$y_{il}^t - y_{il}^{t+1} \geq -p_{il}^t \qquad \forall i,l \in Q, i < l, t \in [m-1], \tag{10.13}$$

and the r's are constrained as

$$y_{is}^t - y_{is}^{t+1} \leq r_{is}^t \qquad \forall i \in Q, t \in [m-1], s \in [N-1], \tag{10.14}$$
$$y_{is}^t - y_{is}^{t+1} \geq -r_{is}^t \qquad \forall i \in Q, t \in [m-1], s \in [N-1]. \tag{10.15}$$

10.1.2. Objective function

Next, we formulate the objective function. The goal of this entire chapter is of course to minimize the variables p and r, as they count the changes in qubit order and the qubits swapping to another computer, respectively. Note that every time two qubits on different computers are swapped, both the corresponding r- and p-variables become 1. Swapping two qubits on different quantum computers should only cost α and not $\alpha+\beta$. We therefore subtract β from the r-term in the objective function. The objective function becomes

$$\min \sum_{t\in[m-1]} \left(\left(\frac{\alpha-\beta}{2} \sum_{\substack{i\in Q \\ s\in[N-1]}} r_{is}^t \right) + \left(\beta \sum_{\substack{i,l\in Q \\ i<l}} p_{il}^t \right) \right), \tag{10.16}$$

where the $(\alpha-\beta)$ term counteracts the extra counting of the SWAP gate with cost α and the factor of one half prevents us from counting the SWAP over the border between computers twice (once for both qubits). Besides that, we sum over the r- and p-variables, since they count the number of inversions between consecutive qubit orders.

10.1.3. Complete model and size

The complete integer linear program combines the objective function and the constraints. The program reads

$$
\begin{aligned}
\min \quad & \sum_{t\in[m-1]} \left(\left(\frac{\alpha-\beta}{2} \sum_{\substack{i\in Q \\ s\in[N-1]}} r_{is}^t \right) + \left(\beta \sum_{\substack{i,l\in Q \\ i<l}} p_{il}^t \right) \right) \\
\text{s.t.} \quad & x_i^t - x_l^t \leq 1 && \forall g_{il}^t \in G \\
& x_i^t - x_l^t \geq -1 && \forall g_{il}^t \in G. \\
& x_i^t - x_l^t \leq M y_{il}^t - 1 && \forall i,l \in Q, i<l, t\in[m] \\
& x_l^t - x_i^t \leq M(1-y_{il}^t) - 1 && \forall i,l \in Q, i<l, t\in[m] \\
& x_i^t - b_s \leq M y_{is}^t - 1 && \forall i \in Q, t\in[m], s\in[N-1] \\
& b_s - x_i^t \leq M(1-y_{is}^t) - 1 && \forall i \in Q, t\in[m], s\in[N-1] \\
& y_{il}^t - y_{il}^{t+1} \leq p_{il}^t && \forall i,l \in Q, i<l, t\in[m-1] \\
& y_{il}^t - y_{il}^{t+1} \geq -p_{il}^t && \forall i,l \in Q, i<l, t\in[m-1] \\
& y_{is}^t - y_{is}^{t+1} \leq r_{is}^t && \forall i \in Q, t\in[m-1], s\in[N-1] \\
& y_{is}^t - y_{is}^{t+1} \geq -r_{is}^t && \forall i \in Q, t\in[m-1], s\in[N-1] \\
& x_i^t \in \cup_{i\in[N]} L_i && \forall i \in Q, t\in[m] \\
& y_{il}^t \in \{0,1\} && \forall i,l \in Q, i<l, t\in[m] \\
& y_{is}^t \in \{0,1\} && \forall i \in Q, t\in[m], s\in[N-1] \\
& r_{is}^t \in \{0,1\} && \forall i \in Q, t\in[m-1], s\in[N-1] \\
& p_{il}^t \in \{0,1\} && \forall i,l \in Q, i<l, t\in[m-1].
\end{aligned}
\tag{10.17}
$$

Here, we let $M = n + \sum_i k_i$. The size of the ILP model scales as a polynomial in the number of qubits, quantum gates and quantum computers in the instance. The number of variables and the number of constraints are both of the order $\mathcal{O}(n^2 m + nMm) = \mathcal{O}(n^2 m)$. The size of the ILP presented in Chapter 5 experiences the same scaling. The programs are closely related. The main difference is in the weighted objective function that incorporates costs corresponding two different types of SWAP operations. Another difference is that we have borders between different quantum computers, of which the order compared to qubits also has to be taken into account.

11

Research Area IV: Celestial Reordering of Qubits in a Distributed Quantum Circuit

This Chapter was written in collaboration with Roy v. Houte.

In this chapter we will introduce the problem of Celestial reordering, which corresponds to research area IV from Table 1.1. In Celestial reordering, given a quantum circuit consisting of qubits, quantum gates acting on the qubits and a number of quantum computers with given capacities, the task is to assign the qubits to the computers in such a way that the number of gate operations on pairs of qubits on different computers is minimized. We assume that the cost of setting up entanglement between two computers is significantly higher than the cost of applying gates within a computer. Therefore, we neglect costs related to gates that are applied on qubits that are located on the same computer.

It is of great importance how the quantum computers are connected in a network. In this section we consider the most straightforward geometries: the completely connected network, the general network, the linear array, the two-dimensional grid and the general grid. For each of the networks, we formalize and visualize the problem, and model it as an integer linear program (ILP). A paper on the results of this chapter has been submitted for possible publication [77].

First we introduce some notation that we will use throughout the chapter.

1. n denotes the total number of qubits in the quantum algorithm. In diagrams, vertices that represent qubits are denoted by circles.
2. M denotes the number of quantum computers. In diagrams, quantum computers are represented by rounded squares.
3. K is used to denote the effective capacity of each quantum computer. That is, the maximum number of qubits that an individual quantum computer can use and store in working memory. This does not include the qubits that are necessary for communication or entanglement swapping. This adds an extra number of qubits per computer, depending on the network architecture.

Suppose we have a quantum algorithm acting on n qubits, that is represented by a series of unitary gates. We allow for unitary operations on single qubits or controlled gates on two qubits. The unitary operations on single qubits will be ignored. All other operations are assumed to be decomposed into this set of gates [63]. One way to ensure sufficient capacity is to take $K \geq \lceil n/M \rceil$ for every computer. If necessary, this quantity may vary per computer as long as the total capacity exceeds n.

11.1. Completely connected network

We start out in the setting where we have all-to-all coupling between the different quantum computers.

Consider the complete graph K_n, where the vertices are labelled $[n] := \{1, \ldots, n\}$ and each vertex corresponds to a qubit in the algorithms. We can count the number of controlled gates that are applied to each pair of

qubits. Similar to the model of single quantum computer global reordering [46], we create a cost function $c : E(K_n) \to \mathbb{Z}_{\geq 0}$ by letting $c_{ij} = c(\{i, j\})$ be the number of controlled gates between qubits i and j. This graph is called the *interaction graph*.

Our goal now is to find an assignment of qubits to computers $f : \{1, \ldots, n\} \to \{1, \ldots, M\}$ such that the total number of controlled gates between all different pairs of computers is minimal.

For a qubit $i \in [n]$ and computer $k \in [M]$ let

$$x_{ik} = \begin{cases} 1 & \text{if qubit } i \text{ is assigned to computer } k \\ 0 & \text{otherwise.} \end{cases} \tag{11.1}$$

For each computer, we thus want to limit the total number of assigned qubits by the computer's total capacity K, so

$$\sum_{i=1}^{n} x_{ik} \leq K, \quad \forall k \in [M]. \tag{11.2}$$

Furthermore, every qubit can be assigned to only one computer, thus

$$\sum_{k=1}^{M} x_{ik} = 1, \quad \forall i \in [n]. \tag{11.3}$$

The objective is

$$\min \sum_{\substack{i,j \in [n] \\ i<j}} c_{ij} \sum_{k \in [M]} \frac{|x_{ik} - x_{jk}|}{2}, \tag{11.4}$$

since for a given $i, j \in [n]$, $i \neq j$. The second summation in the objective is given by

$$\sum_{k \in [M]} \frac{|x_{ik} - x_{jk}|}{2} = \begin{cases} 1 & \text{if qubit } i \text{ and } j \text{ are assigned to different computers} \\ 0 & \text{otherwise.} \end{cases} \tag{11.5}$$

The 2 in the denominator is to compensate for counting twice that a qubit is on a computer where the other qubit is not. This constant can be taken out of the sums.

We can remove the absolute value in the objective by introducing the variable L_{ijk} and add an extra pair of constraints $-L_{ijk} \leq x_{ik} - x_{jk} \leq L_{ijk}$ for every $i, j \in [n], i < j$ and $k \in [M]$. Since any optimal solution will have integer values for L_{ijk}, this variable does not necessarily have to be formulated as integer. This gives us an MILP (mixed integer linear program) of the form

$$\begin{aligned} \min \; & \frac{1}{2} \sum_{\substack{i,j \in [n] \\ i<j}} c_{ij} \sum_{k \in [M]} L_{ijk} \\ \text{s.t.} \; & \sum_{i=1}^{n} x_{ik} \leq K, \quad \forall k \in [M] \\ & \sum_{k=1}^{M} x_{ik} = 1, \quad \forall i \in [n] \\ & \left.\begin{aligned} x_{ik} - x_{jk} &\leq L_{ijk} \\ x_{ik} - x_{jk} &\geq -L_{ijk} \end{aligned}\right\}, \quad \forall i, j \in [n], i < j, \forall k \in [M] \\ & x_{ik} \in \{0, 1\}, \quad \forall i \in [n], k \in [M] \\ & L_{ijk} \in \mathbb{R}, \quad \forall i, j \in [n], i < j, \forall k \in [M]. \end{aligned} \tag{11.6}$$

The total number of integer variables is nM and the total number of continuous variables is $\binom{n}{2}m = n(n-1)m/2$. There are $M + n + n(n-1)M = \mathcal{O}(Mn^2)$ constraints in this problem.

It is possible to extend the celestial reordering model by allowing different capacities of computers. This can easily be done by replacing the capacity constraints by

$$\sum_{i=1}^{n} x_{ik} \leq K_k \quad \forall k \in [M], \tag{11.7}$$

where the capacity K_k is now computer specific.

11.2. General networks of quantum computers

Suppose the network of quantum computers is represented by a connected graph $G = (V, E)$, where quantum computers are represented by nodes. A pair of quantum computers can communicate directly if and only if their corresponding nodes are connected by an edge in the graph. If two quantum computers are not connected directly, we can indirectly connect them via intermediate connections with other computers. We can do this by applying *entanglement swapping*. In this case, we search for the shortest path between the pair of computers.

We let the vertex set $V = [M]$ be labeled by the computers and define $w_{k\ell}$ as the length (i.e. the number of edges) of the shortest path between vertices k and ℓ in G. We can therefore consider the problem on the complete graph K_M with edge weights $w_{k\ell}$ for all $k, \ell \in [M], k \neq \ell$. In Figure 11.1, an example is given for clarification.

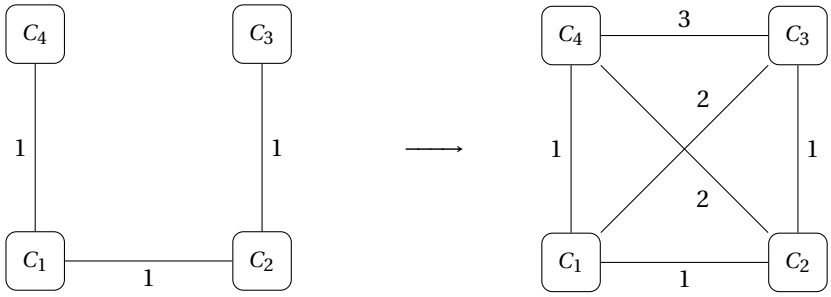


Figure 11.1: The conversion of a general graph to a complete graph with edge weights corresponding to the distance of the shortest path between each pair of nodes. In this example, the shortest path between C_3 and C_4 in the left graph is 3, therefore, the weight on the edge $\{C_3, C_4\}$ in the right graph is equal to 3.

Here, we see that for a network of four computers, we can construct a complete graph where every computer is connected to every other computer. The weights on the edges now indicate the length of the path from one computer to another. Pairs of qubits that are placed on different computers contribute to the costs if they interact with each other. The cost per interaction is equal to the distance between the computers on which the interacting qubits are located, since that counts the number of times that an entangled pair of qubits is required.

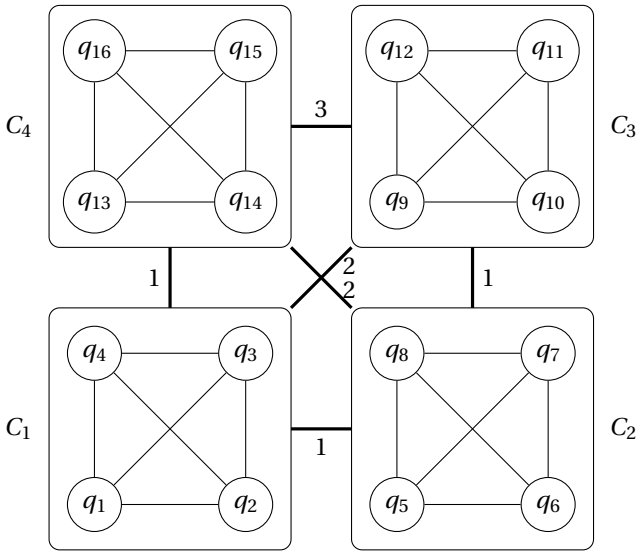


Figure 11.2: A complete graph on a network of four quantum computers, indicated by the C's. The computers each have a capacity of four qubits.

We consider the same decision variables $x_{ik}, i \in [n], k \in [M]$ as in Section 11.1. Our objective will be

$$\min \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij} \sum_{\substack{k,\ell\in[M]\\ k\neq\ell}} w_{k\ell} x_{ik} \cdot x_{j\ell}, \tag{11.8}$$

and since x_{ik} and x_{jl} are both binary, their product is

$$x_{ik} \cdot x_{j\ell} = \begin{cases} 1 & \text{if qubit } i \text{ is on computer } k \text{ and qubit } j \text{ is on computer } \ell \\ 0 & \text{otherwise.} \end{cases} \tag{11.9}$$

The contribution of an assigned pair of qubits depends on two factors: the path length between the computers in the network and the number of interactions in-between the qubits in the algorithms. The product of these quantities is the number of EPR-pairs that is required for this pair of computers.

The constraints are the same as in the original Program 11.6. We thus obtain a quadratic binary optimization program. This quadratic problem has nM variables and $n+M$ constraints.

To transform the quadratic program into an ILP, we introduce a variable $z_{ijk\ell} \in \{0,1\}$ for $i,j \in [n], i<j$ and $k,\ell \in [M], k \neq \ell$, that satisfies the inequality

$$z_{ijk\ell} \geq x_{ik} + x_{j\ell} - 1, \quad \forall i,j \in [n], i<j \text{ and } k,\ell \in [M], k \neq \ell. \tag{11.10}$$

If x_{ik}, $x_{j\ell}$ or both are equal to 0, then $z_{ijk\ell} \geq 0$ and since we are minimizing over an increasing function this yields $z_{ijk\ell} = 0$. Only if $x_{ik} = x_{j\ell} = 1$, then $z_{ijk\ell} = 1$ is required. We are left with the equivalent program

$$\begin{aligned} \min & \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij} \sum_{\substack{k,\ell\in[M]\\ k\neq\ell}} w_{k\ell} z_{ijk\ell} \\ \text{s.t. } & z_{ijk\ell} \geq x_{ik} + x_{j\ell} - 1, \quad \forall i,j \in [n], i<j \text{ and } k,\ell \in [M], k \neq \ell \\ & \sum_{k\in[M]} x_{ik} = 1, \quad \forall i \in [n] \\ & \sum_{i\in[n]} x_{ik} \leq K, \quad \forall k \in [M] \\ & x_{ik} \in \{0,1\}, \quad \forall i \in [n], k \in [M] \\ & z_{ijk\ell} \in \{0,1\}, \quad \forall i,j \in [n], i<j \text{ and } k,\ell \in [M], k \neq \ell. \end{aligned} \tag{11.11}$$

This is an ILP with $nM + \binom{n}{2}M(M-1) = \mathcal{O}(n^2M^2)$ variables and $\mathcal{O}(n^2M^2)$ constraints. Notice that the number of variables and constraints has increased by turning the quadratic program into an ILP.

An interesting question is how much the objective can vary as the capacity K of each computer changes. We can illustrate this with the example of the graphs K_6 and K_4 that are connected by one edge, see Figure 11.3.

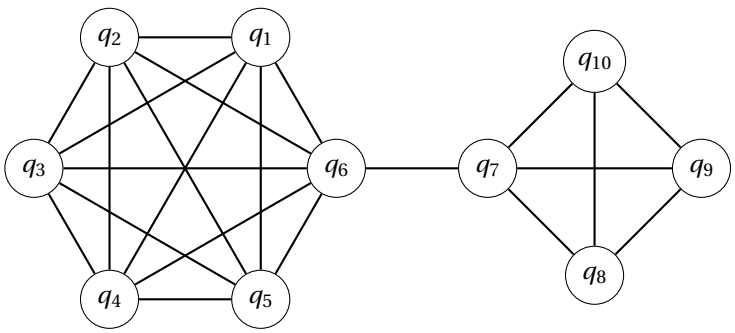


Figure 11.3: Here we see an interaction graph that consists of two complete graphs that are connected to each other by one edge. The graph contains ten qubits. Each edge represents a single interaction between a pair of qubits for some quantum algorithm on ten qubits.

If we have $M = 2$ computers, both with capacity $K = 5$, we are required to make a cut of at least 5 edges. We partition the interaction graph into $\{1,\ldots,5\}$ and $\{6,\ldots,10\}$ forming two computers.

However, if we were somehow able to increase the capacity of both computers to $K = 6$ qubits, we can partition the graph into $\{1,\dots,5\}$ and $\{6,\dots,10\}$. This requires a cut of only one edge. This example and generalizations to more qubits show that the capacity of the computers by a small amount can yield a big difference in the number of EPR pairs required.

11.3. Linear array

In this section, we consider a different network of quantum computers. In this network all computers are arranged on a line, and each one of them is connected to its one or two neighboring computers. This network is a special case of the general network and leads to a reduction in the number of variables and constraints in the resulting model because of the structure in the network.

If we associate the computers with the numbers $\{1,\dots,M\}$ then quantum computer k can only communicate with computers $k-1$ and $k+1$, except at the boundaries. An example of such a network is given in Figure 11.4.

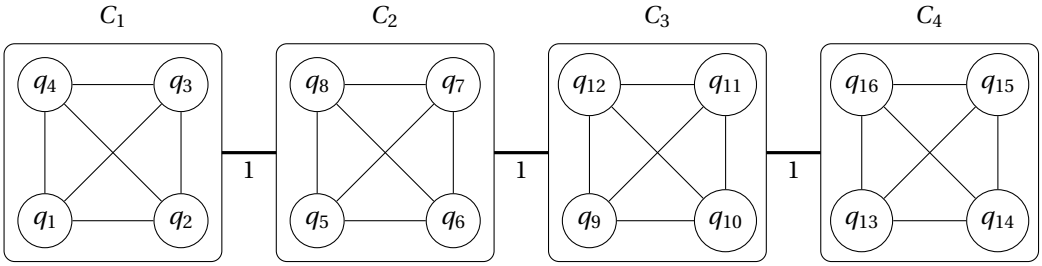


Figure 11.4: A line graph on a network of four quantum computers, indicated by the C's. The computers each have a capacity of four qubits, corresponding to the circular nodes.

This means that if two qubits are located on computer k and ℓ, then applying a two qubit gate requires us to make $|k-\ell|$ non-local interactions by using the computers in-between. This changes the original objective in Equation 11.4 to an objective that takes the distance between computers into account. For a given pair of qubits (i,j) this is given by the equation

$$\left|\sum_{k\in[M]} kx_{ik} - \sum_{k\in[M]} kx_{jk}\right| = \left|\sum_{k\in[M]} k(x_{ik}-x_{jk})\right|. \tag{11.12}$$

Again, we introduce a new variable to encode the absolute value as a linear constraint, analogous to the completely connected network of Section 11.1. The full mixed integer linear program now reads

$$\begin{aligned}
\min\ & \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij}L_{ij} \\
\text{s.t.}\ & \sum_{i=1}^{n} x_{ik} \le K, \quad \forall k\in[M] \\
& \sum_{k=1}^{M} x_{ik} = 1, \quad \forall i\in[n] \\
& \left.\begin{aligned} \sum_{k\in[M]} k(x_{ik}-x_{jk}) &\le L_{ij} \\ \sum_{k\in[M]} k(x_{ik}-x_{jk}) &\ge -L_{ij} \end{aligned}\right\}, \quad \forall i,j\in[n], i<j \\
& x_{ik}\in\{0,1\} \quad \forall i\in[n], k\in[M] \\
& L_{ij}\in\mathbb{R} \quad \forall i,j\in[n], i<j.
\end{aligned} \tag{11.13}$$

This MILP consists of nM integer variables and $\binom{n}{2} = \mathcal{O}(n^2)$ continuous variables and has $n+M+2\binom{n}{2} = \mathcal{O}(n^2+M)$ constraints.

11.4. Two-dimensional grid

In this section we consider a two-dimensional grid as network topology. Such a network allows for more connections between computers and directly extends the linear network of Section 11.3. Nevertheless, this

network also leads to a reduction in the complexity in the assignment of qubits to computers.

We first have to introduce some tools to describe this network. Consider the metric based on the 1-norm[1] defined by

$$d(x,y) = \|x-y\|_1 = \sum_{i=1}^{p} |x_i - y_i|, \quad x, y \in \mathbb{Z}^p. \tag{11.14}$$

We first consider a (square) grid with side length m defined by $G_2 := [m_1] \times [m_2] \subseteq \mathbb{Z}^2$. Thus the number of quantum computers equals $M = m_1 m_2$. We say that two quantum computers are connected if and only if their distance is 1. A small example of such a network is shown in Figure 11.5.

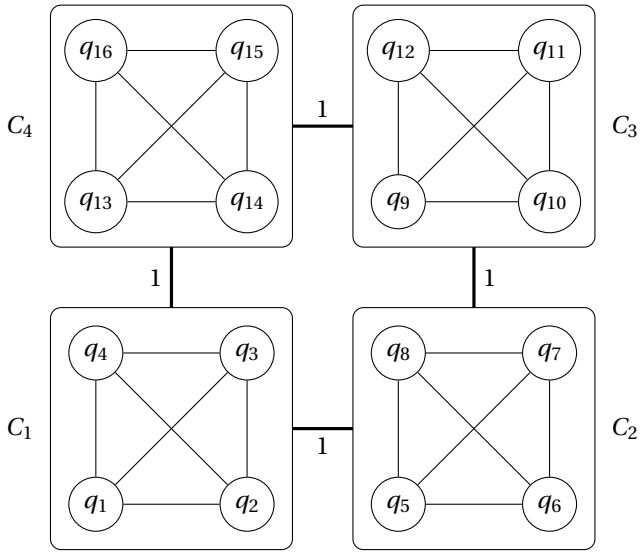


Figure 11.5: The graph of a two-dimensional grid on a network of four quantum computers, indicated by the C's. The computers each have a capacity of four qubits.

For qubit $i \in [n]$ and computer $(u,v) \in G_2$ we let

$$x_{i,uv} = \begin{cases} 1 & \text{if qubit } i \text{ is assigned to position } (u,v) \\ 0 & \text{otherwise.} \end{cases} \tag{11.15}$$

Then, similar to the constraints in Equations 11.2 and 11.3, we have the following constraints:

$$\sum_{i=1}^{n} x_{i,uv} \leq K, \quad \forall (u,v) \in G_2, \tag{11.16}$$

and

$$\sum_{(u,v)\in G_2} x_{i,uv} = 1, \quad \forall i \in [n]. \tag{11.17}$$

Furthermore, the objective is now a weighted sum. The weights are determined by the number of interactions between a pair of qubits. The weighted sum consists of terms given by the distance between computers in the network to which the qubits are assigned. The objective is

$$\begin{aligned} \sum_{\substack{i,j\in[n] \\ i<j}} c_{ij} \|f(i) - f(j)\|_1 = \sum_{\substack{i,j\in[n] \\ i<j}} c_{ij} \Bigg(& \sum_{u\in[m_1]} \left| \sum_{v\in[m_2]} v x_{i,uv} - \sum_{v\in[m_2]} v x_{j,uv} \right| \\ & + \sum_{v\in[m_2]} \left| \sum_{u\in[m_1]} u x_{i,uv} - \sum_{u\in[m_1]} u x_{j,uv} \right| \Bigg). \end{aligned} \tag{11.18}$$

[1] This metric is also called the *taxicab distance* or *Manhattan distance* for its similarity to travelling along the shortest route between two points in the streets of Manhattan, New York.

Again, we introduce new variables to encode the absolute values, this is done by two families $L^{(1)}_{ij,u}$ and $L^{(2)}_{ij,v}$. Analogous to the complete linear network described in Section 11.3, these variables can be relaxed to real numbers. The mixed integer linear program then reads

$$
\begin{aligned}
\min \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij} \left(\sum_{u\in[m_1]} L^{(1)}_{ij,u} + \sum_{v\in[m_2]} L^{(2)}_{ij,v} \right) & \\
\text{s.t.} \sum_{i=1}^{n} x_{i,uv} \le K, \quad \forall\, u\in[m_1], v\in[m_2] & \\
\sum_{u\in[m_1]} \sum_{v\in[m_2]} x_{i,uv} = 1, \quad \forall\, i\in[n] & \\
\left.\begin{aligned}
\sum_{v\in[m_2]} v(x_{i,uv} - x_{j,uv}) \le L^{(1)}_{ij,u} \\
\sum_{v\in[m_2]} v(x_{i,uv} - x_{j,uv}) \ge -L^{(1)}_{ij,u}
\end{aligned}\right\} \quad \forall\, u\in[m_1] & \\
\left.\begin{aligned}
\sum_{u\in[m_1]} u(x_{i,uv} - x_{j,uv}) \le L^{(2)}_{ij,v} \\
\sum_{u\in[m_1]} u(x_{i,uv} - x_{j,uv}) \ge -L^{(2)}_{ij,v}
\end{aligned}\right\} \quad \forall\, u\in[m_1] & \quad \Bigg\} \quad \forall\, i,j\in[n], i<j \\
x_{i,uv}\in\{0,1\}, \quad \forall\, i\in[n], u\in[m_1], v\in[m_2] & \\
L^{(1)}_{ij,u}, L^{(2)}_{ij,v} \in \mathbb{R}, \quad \forall\, u\in[m_1], v\in[m_2], \forall\, i,j\in[n], i<j. &
\end{aligned}
\tag{11.19}
$$

This MILP has $nm_1m_2 = nM$ integer variables and $\binom{n}{2}(m_1+m_2) = \mathcal{O}(n^2(m_1+m_2))$ continuous variables. The program contains $m_1m_2 + n + 2\binom{n}{2}(m_1+m_2) = \mathcal{O}(n^2(m_1+m_2)+M)$ constraints. If the grid sizes are similar up to a constant, then $m_1 = \Theta(m_2)$. Furthermore, if the capacity of each computer is fixed and the least number of computers is used, then $n = \mathcal{O}(M)$, then the number of constraints is $\mathcal{O}(M^{2.5})$.

11.5. General grid

The two-dimensional network of Section 11.4 was a generalisation of the linear network of Section 11.3. Since the 1-norm allows for generalisation to any finite dimensional lattice, this section describes the most general case for grids.

We assume the dimensions of the p-dimensional grid are the same to provide a clearer description. However, similar to the two-dimensional grid, it is possible to use grids of different spatial proportions. Let $G_p = \underbrace{[m]\times\cdots\times[m]}_{p \text{ times}}$, and for $u = (u_1,\dots,u_{p-1}) \in G_{p-1}$, $r\in[d]$, $v\in[m]$ define

$$
u \oplus_r v = (u_1,\dots,u_{r-1},v,u_r,\dots,u_{p-1}) \in G_p, \tag{11.20}
$$

that is, in the integer string u we insert the number v at place r.

The general objective now becomes

$$
\sum_{\substack{i,j\in[n]\\ i<j}} c_{ij} \left(\sum_{r\in[p]} \left(\sum_{u\in G_{p-1}} \left| \sum_{v\in[m]} v(x_{i,u\oplus_r v} - x_{j,u\oplus_r v}) \right| \right) \right). \tag{11.21}
$$

Here, we can again introduce a family of variables $L^{(r)}_{ij,u}$ for all $i,j\in[n]$, $i<j$, $u\in G_{d-1}$ and $r\in[m]$ to linearize

the absolute value. This gives us the MILP

$$
\begin{aligned}
\min \; & \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij}\left(\sum_{r\in[p]}\left(\sum_{u\in G_{p-1}} L^{(r)}_{ij,u}\right)\right) \\
\text{s.t.} \; & \sum_{i\in[n]} x_{i,\omega} \le K, \quad \forall \omega \in G_p \\
& \sum_{\omega\in G_p} x_{i,\omega} = 1, \quad \forall i \in [n] \\
& \left.\begin{aligned} \sum_{v\in[m]} u(x_{i,u\oplus_r v} - x_{j,u\oplus_r v}) &\le L^{(r)}_{ij,u} \\ \sum_{v\in[m]} u(x_{i,u\oplus_r v} - x_{j,u\oplus_r v}) &\ge -L^{(r)}_{ij,u} \end{aligned}\right\} \quad \forall u \in G_{p-1}, r\in[p], i,j\in[n], i<j \\
& x_{i,\omega} \in \{0,1\}, \quad \forall i \in [n], \omega \in G_p \\
& L^{(r)}_{ij,u} \in \mathbb{R}, \quad \forall u \in G_{p-1}, r\in[p], i,j\in[n], i<j.
\end{aligned}
\tag{11.22}
$$

The number of quantum computers in this network is $M = m^p$. This program contains nM integer variables and $\binom{n}{2} p m^{p-1} = \mathcal{O}(n^2 p M^{(p-1)/p})$ continuous variables. Furthermore, there are $m^p + n + 2\binom{n}{2} p m^{p-1} = \mathcal{O}(M + n^2 p M^{(p-1)/p})$ constraints. For $p = 2$ we indeed get the result of the previous section.

12

Conclusion and Future Research

In this thesis, we analyze the NNC problem on a variety of qubit architectures. The main focus is on exact solution methods for the local reordering problem on a single quantum computer.

12.1. Conclusion

We formulated a research question and decomposed it into three sub-questions. We first answer the three sub-questions, and provide an answer to the main research question afterwards.

1. *How can we model the nearest neighbor compliance problem such that the size of the model scales polynomially instead of exponentially, while retaining exact solutions?*
   The integer linear program (ILP) introduced in Chapter 5 uses big-M type constraints to implicitly calculate the Kendall tau metric. This circumvents explicitly listing all permutations of the qubits together with the related costs in terms of SWAP gate use. The proposed model is suitable for exact solution methods, and larger circuits can be evaluated than in the current literature. The number of variables and constraints in the ILP is $\mathcal{O}(n^2 m)$, where n denotes the number of qubits and m denotes the number of two-qubit quantum gates after gate decomposition. This result is submitted for possible publication.

   An attempt was made to model the nearest neighbor compliance problem more efficiently by incorporating concepts of graph theory. Here, the problem was reduced to a single pair shortest path problem on a Cayley graph of the permutation group, called the adjacent transposition graph. This model does not scale polynomially in the number of qubits and quantum gates in the considered quantum circuit. It might be possible to model the nearest neighbor compliance problem differently, as a variant on the traveling salesman problem for example, such that the number of edges and nodes does scale polyomially in the number of qubits and quantum gates.

2. *How does the nearest neighbor compliance problem for two- and three-dimensional architectures relate to the nearest neighbor compliance problem on a linear array?*
   We have formulated an integer linear program to exactly evaluate the two- and three-dimensional grid variants of the nearest neighbor compliance problem. The number of variables scales in both cases as $\mathcal{O}(n^4 m)$, and the number of constraints scales in both cases as $\mathcal{O}(n^3 m)$, where n denotes the number of qubits and m denotes the number of two-qubit gates after gate decomposition. This is clearly worse than was the case for the linear array. For a fixed quantum ciruit, when considering a two- or three-dimensional grid, the optimal objective value is always bounded from above by the objective value in the case of the linear array qubit architecture. A solution to the NNC problem for the linear array can always be transformed to a solution of the NNC problem on a grid, following the proof of Theorem 6.4. This solution could be used as a starting point for the Branch & Bound search in the two-dimensional and three-dimensional cases.

3. *What are the main advantages and disadvantages of exact solution methods for the nearest neighbor compliance problem?*
   The foremost advantage of an exact solution method in the case of the nearest neighbor compliance

problem, is that optimal SWAP gate insertion in small circuits is carried through to bigger circuits in which the small circuits are often incorporated. In the bigger circuits, there might be a better way to combine the small circuits, but it is at least compartment-wise optimal. Another benefit of an exact solution approach is that heuristic solutions can now be evaluated and compared to exact solutions for the benchmark instances that we have analyzed. There are several disadvantages to our approach due to the restrictive assumptions and simplifications that we have made: 1) The gate decomposition method is fixed for each gate-type, no optimization is done in this area, 2) The gate sequence is fixed while, under certain commutative conditions, it can be allowed and beneficial to change the gate sequence, 3) The depth of a circuit, which is related to the running time on a quantum computer and which is limited by the coherence times of the qubits, is not taken into consideration directly.

Now that each of the sub-questions have been answered, we provide an answer to the main research question.

How can we effectively convert quantum circuits by inserting SWAP gates, such that the converted circuits comply with the nearest neighbor constraints? The graph theoretical model that we presented mostly functions as an intuitive introduction to the problem and is unsuited to evaluate circuits with even ten qubits. The framework of integer linear programming enables the implicit calculation of the Kendall tau distance in the case of a linear array qubit architecture. The resulting model has the capacity to evaluate larger benchmark instances to the nearest neighbor compliance problem with an exact solution approach than previously possible. In the case where qubits are placed on two- or three-dimensional grids, the proposed models do not incorporate the implicit calculation of the number of inserted SWAP gates and require more variables and constraints. Only taking into account one manner of decomposing multi-qubit gates and a fixed gate sequence, we can hardly claim that our models provide circuits that are optimal in all regards. However, the results show that in terms of circuit size, the proposed methods outperform current exact solution approaches in the literature.

Besides the main focus of the thesis, a few other topics have been analyzed. The remaining research areas as indicated in Table 1.1 were each analyzed in Part II of this thesis. We provided a literature review for the Global Reordering problem. The other two settings that we have considered in which calculation overhead is minimized are about distributed quantum computing. We have formulated the nearest neighbor compliance problem on a linear array of multiple quantum computers as an ILP. Here, we distinguish between SWAP operations that swap qubits between different computers from SWAP operations that swap qubits that are on the same computer. We have also introduced the Celestial Reordering problem, which concerns the minimization of interactions between different quantum computer by considering the initial qubit placement. In Celestial Reordering, different computer architectures are considered, and an ILP formulation is provided for each architecture. The results on the topics regarding distributed quantum computing, have been submitted for possible publication.

12.2. Future Research

There are several promising areas of research as to how this work can be extended. In this section we suggest several topics that could lead to potentially interesting theory.

We expect that the most easily obtainable results can be found by looking into patterns regarding quantum circuits that can have a variable input in the number of qubits. To improve the results for a widely used circuit, the Quantum Fourier Transform (QFT) for example, one could compare the evaluations on different sizes of the circuit. Right now, we have evaluated the QFT circuit from three up to ten qubits. Finding a pattern between the optimal SWAP insertions for the different sizes of the QFT is highly likely and the results can likely be extended to the more general case of n qubits.

Another interesting research area would be to implement both a Breadth First Search and the A^* algorithm to see how the single pair shortest path formulation performs in combination with the heuristic. The most promising part of this area would be to evaluate quantum circuits that consist of few qubits and a lot of quantum gates. Another approach would be to generate the adjacent transposition graph as it is being explored by a shortest path algorithm. In that case, not all of the $\mathcal{O}(n!m)$ nodes and edges have to be generated explicitly, but only those considered for the route. Since the number of SWAP gates in most benchmark instances is very limited, the number of explored nodes and edges should similarly remain small relative to the size of the entire graph.

Similarly, one could opt for a reduction to the Traveling Salesman problem (TSP) or a variant thereof. We

expect that the size of the graph in this case, could scale linearly in the number of qubits and gates. A lot of work was already done to solve the TSP efficiently. The available results on the TSP could then be adapted to solve the Nearest Neighbor Compliance problem more efficiently than current attempts.

One of the perhaps more challenging directions of research is on the theoretical side. Proving that the Nearest Neighbor Compliance problem as formulated in this work is NP-complete, would be an achievement. We suggest to reduce the NP-complete Subgroup Distance problem to the Nearest Neighbor Compliance problem by introducing quantum gates at the start of a circuit that limit the qubit order to a certain subgroup of the symmetric group and working from there.

The most interesting and perhaps most challenging research topic regards a metric alike the Kendall tau metric, but for two- and three- dimensional grid qubit architectures. If such a metric can be implicitly calculated in an integer linear program, the scaling of the size of the proposed models for the two- and three-dimensional cases can be reduced by a factor of up to n^2 variables. A resulting model can be used to evaluate larger benchmark instances that can in turn be used to evaluate heuristic methods.

Deviating from the main focus, and redirecting attention to the related research areas, which were discussed in Part II, could also prove worthwhile. Not a lot of attention was focused on the areas involving discrete quantum computing. Simple heuristics and naive implementations of exact algorithms could already help to indicate the difficulty of the proposed problems. The problem of Celestial reordering can potentially benefit from the existing literature on clustering algorithms.

Bibliography

[1] M. Gh. AlFailakawi, L. AlTerkawi, I. Ahmad, and S. Hamdan. Line ordering of reversible circuits for linear nearest neighbor realization. *Quantum Inf. Process.*, 12(10):3319–3339, October 2013.

[2] M. Gh. AlFailakawi, I. Ahmad, and S. Hamdan. Harmony-search algorithm for 2d nearest neighbor quantum circuits realization. *Expert Syst. with Appl.*, 61:16–27, November 2016.

[3] N. Alhagi. Synthesis of Reversible Functions Using Various Gate Libraries and Design Specifications. Technical report, Portland State University, January 2000.

[4] J. M. Amini, H. Uys, J. H. Wesenberg, S. Seidelin, J. Britton, J. J. Bollinger, D. Leibfried, C. Ospelkaus, A. P. VanDevender, and D. J. Wineland. Toward scalable ion traps for quantum information processing. *New J. Phys.*, 12(3):033031, March 2010.

[5] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. W. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52(5):3457–3467, November 1995.

[6] J. J. Berwald. The Mathematics of Quantum-Enabled Applications on the D-Wave Quantum Computer. *Not. Amer. Math. Soc.*, 66(06):1, June 2019.

[7] A. Bhattacharjee, C. Bandyopadhyay, R. Wille, R. Drechsler, and H. Rahaman. A Novel Approach for Nearest Neighbor Realization of 2d Quantum Circuits. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 305–310, Hong Kong, July 2018. IEEE. ISBN 978-1-5386-7099-6.

[8] A. Bhattacharjee, C. Bandyopadhyay, R. Wille, R. Drechsler, and H. Rahaman. Improved Look-Ahead Approaches for Nearest Neighbor Synthesis of 1d Quantum Circuits. In *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, pages 203–208, Delhi, NCR, India, January 2019. IEEE. ISBN 978-1-72810-409-6.

[9] É. Bonnet, T. Miltzow, and P. Rzążewski. Complexity of Token Swapping and Its Variants. *Algorithmica*, 80(9):2656–2682, September 2018.

[10] A. Botea, A. Kishimoto, and R. Marinescu. On the Complexity of Quantum Circuit Compilation. page 5, 2018.

[11] H. Buhrman and H. Röhrig. Distributed quantum computing. In *Mathematical Foundations of Computer Science 2003*, pages 1–20, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45138-9.

[12] R. E. Burkard, E. Dragoti-Cela, P. M. Pardalos, and L. S. Pitsoulis. The quadratic assignment problem. *Handbook of Combinatorial Optimization*, pages 241–337, 1998.

[13] D. Castelvecchi. IBM's quantum cloud computer goes commercial : Nature News & Comment. *Nature*, 543:159, March 2017.

[14] T. M. Chan and M. Pătraşcu. Counting Inversions, Offline Orthogonal Range Counting, and Related Problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 161–173. Society for Industrial and Applied Mathematics, January 2010. ISBN 978-0-89871-701-3 978-1-61197-307-5.

[15] X. Cheng, Z. Guan, and W. Ding. Mapping from multiple-control Toffoli circuits to linear nearest neighbor quantum circuits. *Quantum Inf. Process.*, 17(7):169, July 2018.

[16] B.-S. Choi and R. Van Meter. An $\Theta(\sqrt{n})$-depth Quantum Adder on a 2d NTC Quantum Computer Architecture. *J. Emerg. Technol. Comput. Syst.*, 8(3):1–22, August 2012.

[17] A. Cornelissen. Quantum gradient estimation and its application to quantum reinforcement learning. Master's thesis, 2018.

[18] G. Cornuéjols and M. Dawande. A Class of Hard Small 0-1 Programs. *Inf. J. on Comput.*, 11(2):205–210, May 1999.

[19] V. S. Denchev and G. Pandurangan. Distributed quantum computing: A new frontier in distributed systems or science fiction? *SIGACT News*, 39(3):77–95, September 2008.

[20] S. J. Devitt, A. G. Fowler, A. M. Stephens, A. D. Greentree, L. C. L. Hollenberg, W. J. Munro, and K. Nemoto. Architectural design for a topological cluster state quantum computer. *New J. Phys.*, 11(8):083032, August 2009.

[21] R. Diestel. *Graph theory*. Number 173 in Graduate texts in mathematics. Springer, New York, 2nd ed edition, 2000. ISBN 978-0-387-95014-3 978-0-387-98976-1.

[22] J. Ding and S. Yamashita. Exact Synthesis of Nearest Neighbor Compliant Quantum Circuits in 2d architecture and its Application to Large-scale Circuits. *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst.*, pages 1–1, 2019.

[23] D. P. DiVincenzo and IBM. The Physical Implementation of Quantum Computation. *Fortschr. der Phys.*, 48(9-11):771–783, September 2000.

[24] D. P. DiVincenzo and F. Solgun. Multi-qubit parity measurement in circuit quantum electrodynamics. *New J. Phys.*, 15(7):075001, July 2013.

[25] G. W. Dueck, A. Pathak, M. M. S. Rahman, A. Shukla, and A. Banerjee. Optimization of Circuits for IBM's five-qubit Quantum Computers. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 680–684, August 2018.

[26] A. Einstein, M. Born, and H. Born. Born-Einstein letters. 1971.

[27] A. Farghadan and N. Mohammadzadeh. Mapping quantum circuits on 3d nearest-neighbor architectures. *Quantum Sci. Technol.*, 4(3):035001, April 2019.

[28] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg. Implementation of Shor's Algorithm on a Linear Nearest Neighbour Qubit Array. *arXiv:quant-ph/0402196*, February 2004. arXiv: quant-ph/0402196.

[29] A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg. Quantum-error correction on linear-nearest-neighbor qubit arrays. *Phys. Rev. A*, 69(4), April 2004.

[30] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, January 1979. ISBN 978-0-7167-1045-5.

[31] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact Multiple-Control Toffoli Network Synthesis With SAT Techniques. *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst.*, 28(5):703–715, May 2009.

[32] L. K. Grover. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Phys. Rev. Lett.*, 79(2): 325–328, July 1997.

[33] W. Hattori and S. Yamashita. Quantum Circuit Optimization by Changing the Gate Order for 2d Nearest Neighbor Architectures. In *Revers. Comput.*, Lecture Notes in Computer Science, pages 228–243. Springer International Publishing, 2018. ISBN 978-3-319-99498-7.

[34] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenberg, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson. Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres. *Nat.*, 526(7575):682–686, October 2015.

[35] D. A. Herrera-Martí, A. G. Fowler, D. Jennings, and T. Rudolph. Photonic implementation for the topological cluster-state quantum computer. *Phys. Rev. A*, 82(3):032332, September 2010.

[36] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima. An efficient conversion of quantum circuits to a linear nearest neighbor architecture. *Quantum Inf. and Comput.*, 11(1&2):142–166, 2011.

[37] F. J. Humphreys and M. Hatherly. *Recrystallization and Related Annealing Phenomena*. Elsevier, December 2012. ISBN 978-0-08-098388-2. Google-Books-ID: Kt11V4m2bqEC.

[38] T. Itoko, R. Raymond, T. Imamichi, A. Matsuo, and A. W. Cross. Quantum circuit compilers using gate commutation rules. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference on - ASPDAC '19*, pages 191–196, Tokyo, Japan, 2019. ACM Press. ISBN 978-1-4503-6007-4.

[39] M. R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985.

[40] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto. Layered Architecture for Quantum Computing. *Phys. Rev. X*, 2(3):031007, July 2012.

[41] L. Kaufman and F. Broeckx. An algorithm for the quadratic assignment problem using Bender's decomposition. *Eur. J. of Oper. Res.*, 2(3):207–211, May 1978.

[42] J. Kawahara, T. Saitoh, and R. Yoshinaka. The Time Complexity of the Token Swapping Problem and Its Parallel Variants. In *WALCOM: Algorithms and Computation*, Lecture Notes in Computer Science, pages 448–459. Springer International Publishing, 2017. ISBN 978-3-319-53925-6.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Sci.*, 220(4598): 671–680, May 1983.

[44] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition*, volume 3. 2nd edition, 1974.

[45] P. Kok and S. L. Braunstein. Entanglement swapping as event-ready entanglement preparation. *Fortschr. der Phys.*, 48(5-7):553–557, 2000.

[46] A. Kole, K. Datta, I. Sengupta, and R. Wille. Towards a cost metric for nearest neighbor constraints in reversible circuits. In *International Conference on Reversible Computation*, pages 273–278. Springer, 2015.

[47] A. Kole, K. Datta, I. Sengupta, and R. Wille. Towards a Cost Metric for Nearest Neighbor Constraints in Reversible Circuits. *Rev. Comput.*, 9138:273–278, 2015.

[48] A. Kole, K. Datta, and I. Sengupta. A Heuristic for Linear Nearest Neighbor Realization of Quantum Circuits by SWAP Gate Insertion Using$N$-Gate Lookahead. *IEEE J. on Emerg. and Sel. Top. in Circuits and Syst.*, 6(1):62–72, March 2016.

[49] A. Kole, K. Datta, and I. Sengupta. A New Heuristic for N -Dimensional Nearest Neighbor Realization of a Quantum Circuit. *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst.*, 37(1):182–192, January 2018.

[50] M. Kumph, M. Brownnutt, and R. Blatt. Two-dimensional arrays of radio-frequency ion traps with addressable interactions. *New J. Phys.*, 13(7):073043, July 2011.

[51] S. Lakshmivarahan, J.-S. Jwo, and S. K. Dhall. Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey. *Parallel Comput.*, 19(4):361–407, April 1993.

[52] C. Lin, S. Sur-Kolay, and N. K. Jha. PAQCS: Physical Design-Aware Fault-Tolerant Quantum Circuit Synthesis. *IEEE Trans. on Very Large Scale Int. Syst.*, 23(7):1221–1234, July 2015.

[53] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Experimental comparison of two quantum computing architectures. *Proc. Natl. Acad. Sci.*, 114(13):3305–3310, March 2017.

[54] A. Lye, R. Wille, and R. Drechsler. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In *The 20th Asia and South Pacific Design Automation Conference*, pages 178–183, January 2015.

[55] I. L. Markov and M. Saeedi. Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation. *arXiv:1202.6614 [quant-ph]*, February 2012. arXiv: 1202.6614.

[56] D. Maslov, C. Young, D.M. Miller, and G.W. Dueck. Quantum Circuit Simplification Using Templates. In *Design, Automation and Test in Europe*, pages 1208–1213, Munich, Germany, 2005. IEEE.

[57] A. Matsuo and S. Yamashita. Changing the Gate Order for Optimal LNN Conversion. In *Reversible Computation*, Lecture Notes in Computer Science, pages 89–101. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29517-1.

[58] C. C. McGeoch. *Adiabatic quantum computation and quantum annealing: theory and practice*. Number 8 in Synthesis lectures on quantum computing. Morgan & Claypool, San Rafael, Calif., 2014. ISBN 978-1-62705-335-8 978-1-62705-336-5.

[59] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *J. Assoc. for Comput. Mach.*, 7(4):326–329, October 1960.

[60] G. E. Moore. Cramming More Components Onto Integrated Circuits. *Proc. IEEE*, 86(1):82–85, January 1998.

[61] J. Mulderij, K. I. Aardal, I. Chiscop, and F. Phillipson. A polynomial size model with implicit swap gate counting for exact qubit reordering. submitted, 2019.

[62] N. H. Nickerson, Y. Li, and S.C. Benjamin. Topological quantum computing with a very noisy network and local error rates approaching one percent. *Nat. Commun.*, 4(1), December 2013.

[63] M. A. Nielsen and I. Chuang. Quantum Computation and Quantum Information. *Am. J. of Phys.*, 70(5): 558–559, May 2002.

[64] M. Ohliger and J. Eisert. Efficient measurement-based quantum computing with continuous-variable systems. *Phys. Rev. A*, 85(6):062318, June 2012. arXiv: 1112.2641.

[65] M. Pedram and A. Shafaei. Layout Optimization for Quantum Circuits with Linear Nearest Neighbor Architectures. *IEEE Circuits and Syst. Mag.*, 16(2):62–74, 2016.

[66] P. Pham and K. M. Svore. A 2d Nearest-Neighbor Quantum Architecture for Factoring in Polylogarithmic Depth. *arXiv:1207.6655 [quant-ph]*, July 2012. arXiv: 1207.6655.

[67] QuTech. Quantum Inspire Home. URL `https://www.quantum-inspire.com/reference/`.

[68] M. Saeedi, R. Wille, and R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Inf. Process.*, 10(3):355–377, June 2011.

[69] A. Shafaei, M. Saeedi, and M. Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, May 2013.

[70] A. Shafaei, M. Saeedi, and M. Pedram. Qubit placement to minimize communication overhead in 2d quantum architectures. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 495–500, January 2014.

[71] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, 1994. IEEE Comput. Soc. Press. ISBN 978-0-8186-6580-6.

[72] R. R. Shrivastwa, K. Datta, and I. Sengupta. Fast Qubit Placement in 2d Architecture Using Nearest Neighbor Realization. In *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 95–100, December 2015.

[73] M. Y. Siraichi, V. F. dos Santos, S. Collange, and F. M. Q. Pereira. Qubit Allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, pages 113–125, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5617-6. event-place: Vienna, Austria.

[74] Y. Takahashi, N. Kunihiro, and K. Ohta. The Quantum Fourier Transform on a Linear Nearest Neighbor Architecture. *Quantum Info. Comput.*, 7(4):383–391, May 2007.

[75] Y. Tan, X. Cheng, Z. Guan, Y. Liu, and H. Ma. Multi-strategy based quantum cost reduction of linear nearest-neighbor quantum circuit. *Quantum Inf. Process.*, 17(3):61, January 2018.

[76] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.

[77] R. van Houte, J. Mulderij, T. Attema, I. Chiscop, and F. Phillipson. Mathematical formulation of quantum circuit design problems in networks of quantum computers. submitted, 2019.

[78] R. Versluis, S. Poletto, N. Khammassi, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo. Scalable quantum circuit and control for a superconducting surface code. *Phys. Rev. Applied*, 8(3):034021, September 2017. arXiv: 1612.08208.

[79] S. Wehner, D. Elkouss, and R. Hanson. Quantum internet: A vision for the road ahead. *Sci.*, 362(6412), 2018.

[80] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: An Online Resource for Reversible Functions and Reversible Circuits. In *38th International Symposium on Multiple Valued Logic (ismvl 2008)*, pages 220–225, May 2008.

[81] R. Wille, A. Lye, and R. Drechsler. Considering nearest neighbor constraints of quantum circuits at the reversible circuit level. *Quantum Inf. Process.*, 13(2):185–199, February 2014.

[82] R. Wille, A. Lye, and R. Drechsler. Exact Reordering of Circuit Lines for Nearest Neighbor Quantum Architectures. *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst.*, 33(12):1818–1831, December 2014.

[83] R. Wille, O. Keszocze, M. Walter, P. Rohrs, A. Chattopadhyay, and R. Drechsler. Look-ahead schemes for nearest neighbor optimization of 1d and 2d quantum circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 292–297, Macao, Macao, January 2016. IEEE. ISBN 978-1-4673-9569-4.

[84] R. Wille, L. Burgholzer, and A. Zulehner. Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations. In *Proceedings of the 56th Annual Design Automation Conference 2019 on - DAC '19*, pages 1–6, Las Vegas, NV, USA, 2019. ACM Press. ISBN 978-1-4503-6725-7.

[85] L. A. Wolsey. *Integer programming*. Wiley, 1998. ISBN 978-0471283669.

[86] K. Yamanaka, E. D. Demaine, T. Ito, J. Kawahara, M. Kiyomi, Y. Okamoto, T. Saitoh, A. Suzuki, K. Uchizawa, and T. Uno. Swapping Labeled Tokens on Graphs. In *Fun with Algorithms*, volume 8496, pages 364–375. Springer International Publishing, Cham, 2014. ISBN 978-3-319-07889-2 978-3-319-07890-8.

[87] N. Y. Yao, Z.-X. Gong, C. R. Laumann, S. D. Bennett, L.-M. Duan, M. D. Lukin, L. Jiang, and A. V. Gorshkov. Quantum Logic between Remote Quantum Registers. *Phys. Rev. A*, 87(2):022306, February 2013. arXiv: 1206.0014.

[88] A. Yimsiriwattana and S. J. Lomonaco Jr. Distributed quantum computing: A distributed shor algorithm. In *Quantum Information and Computation II*, volume 5436, pages 360–372. International Society for Optics and Photonics, 2004.

[89] A. Zulehner, H. Bauer, and R. Wille. Evaluating the Flexibility of A* for Mapping Quantum Circuits. In *Rev. Comput.*, volume 11497, pages 171–190. Springer International Publishing, Cham, 2019. ISBN 978-3-030-21499-9 978-3-030-21500-2.

[90] A. Zulehner, A. Paler, and R. Wille. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst.*, 38(7):1226–1236, July 2019.