THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Quantum Computing for Airline Planning and Operations

MARIKA SVENSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

Quantum Computing for Airline Planning and Operations

Marika Svensson

This thesis has been prepared using LaTeX.

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden, September 2024

To my grandfather, Professor Emeritus Lennart Pålsson

Abstract

Classical algorithms and mathematical optimization techniques have been used extensively by airlines to optimize their profit and ensure that regulations are followed. In this thesis, we explore which role quantum algorithms can have for airlines. Specifically, we have considered the two quantum optimization algorithms; the Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing (QA). We present a heuristic that integrates these quantum algorithms into the existing classical algorithm, which is currently employed to solve airline planning problems in a state-of-the-art commercial solver. We perform numerical simulations of QAOA circuits and find that linear and quadratic algorithm depth in the input size can be required to obtain a one-shot success probability of 0.5. Unfortunately, we are unable to find performance guarantees. Finally, we perform experiments with D-wave's newly released QA machine and find that it outperforms 2000Q for most instances.

Keywords: discrete optimization, quantum approximate optimization algorithm, quantum annealing, airline scheduling, column generation

List of Publications

This thesis is based on the following publications:

[A] **Marika Svensson**, Martin Andersson, Mattias Grönkvist, Pontus Vikstål, Devdatt Dubhashi, Giulia Ferrini, and Göran Johansson, “A Hybrid Quantum-Classical Heuristic to solve large-scale Integer Linear Programs”. arXiv:2103.15433 (accepted to Physical Review Applied).

[B] Dennis Willsch, Madita Willsch, Carlos D. Gonzalez Calaza, Fengping Jin, Hans De Raedt, **Marika Svensson**, and Kristel Michielsen, “Benchmarking Advantage and D-Wave 2000Q quantum annealers with exact cover problems”. Quantum Inf Process 21, 141 (2022).

[C] Pontus Vikstål, Mattias Grönkvist, **Marika Svensson**, Martin Andersson, Göran Johansson, and Giulia Ferrini, “Applying the Quantum Approximate Optimization Algorithm to the Tail Assignment Problem”. Phys. Rev. Applied 14, 034009 (2020).

[D] Andreas Bengtsson, Pontus Vikstål, Christopher Warren, **Marika Svensson**, Xiu Gu, Anton Frisk Kockum, Philip Krantz, Christian Križan, Daryoush Shiri, Ida-Maria Svensson, Giovanna Tancredi, Göran Johansson, Per Delsing, Giulia Ferrini, and Jonas Bylander, “Improved success probability with greater circuit depth for the quantum approximate optimization algorithm”. Phys. Rev. Applied 14, 034010 (2020).

Other related work

[1] Rishi Sreedhar, Pontus Vikstål, **Marika Svensson**, Andreas Ask, Göran Johansson, and Laura García-Álvarez, “The Quantum Approximate Optimization Algorithm performance with low entanglement and high circuit depth”. arXiv:2207.05612.

Acknowledgments

I would first like to thank my advisors at Jeppesen, Mattias Grönkvist, and Martin Andersson for their continuous support and many useful discussions. I would also like to thank Andreas Gunnarson for his genuine interest in this project, the Pairing team, and my manager Åse Kyttä for all the support given.

I would also like to thank my advisor at Chalmers University of Technology, Göran Johansson, for the vital aspects of physics, guidance within academics, and his newfound interest in integer programming. Then, I would like to thank the assistant advisor, Devdatt Dubhashi, for his support around computational complexity and, generally, theoretical computer science topics.

Furthermore, I would like to thank Pontus Vikstål for collaboration and discussions regarding simulations of quantum computers, and finally, Giulia Ferrini for exciting discussions and support in this truly interdisciplinary field.

Finally, I would like to express my gratitude to all the members of the Division of Computer Science and Artificial Intelligence, as well as the research group of Applied Quantum Physics, for providing such a pleasant working environment.

Contents

Part I

Overview

CHAPTER 1

Introduction

Airlines have used classical algorithms and mathematical optimization [1] techniques extensively for decades [2]. In this thesis, we explore which role quantum algorithms can have in this context. In the following sections, we give a brief overview of the airline industry, airline scheduling problems, and quantum computing.

1.1 The Airline Scheduling Process

Airlines operate within an industry that is highly competitive with large operational costs. In particular, the largest costs are related to fuel consumption and crew. The airlines are furthermore governed by many operational rules imposed by aviation authorities and unions. It is also common that airlines themselves have internal rules that need to be respected. Additionally, airlines also must deal with uncertainties due to weather conditions and other disruptions. These characteristics force airlines to carefully schedule their flights, crew, and aircraft to maximize revenue and minimize cost.

Scheduling Problems

An airline has two significant decisions before scheduling [3], namely the fleet size and structure and which routes to cover. Typically an airline begins by determining the fleet, i.e., how many aircraft of each fleet type should exist and how many aircraft there are in total. Given that the set of aircraft is determined, the airline can consider which origins and destinations they wish to cover, i.e., route planning. Generally, after these planning stages are completed, the airline can consider the scheduling of flights, aircraft, and crew.

The scheduling problem [4], [5] has historically been divided into subproblems Flight Scheduling, Fleet Assignment, Crew Scheduling, Tail Assignment, and lastly Recovery Planning. Commonly, Crew Scheduling is further partitioned into Crew Pairing and Crew Rostering. In this setting, the output of a subproblem is input to the following subproblem and solved sequentially; see Fig. 1.1. However, by considering the subproblems individually, the over-

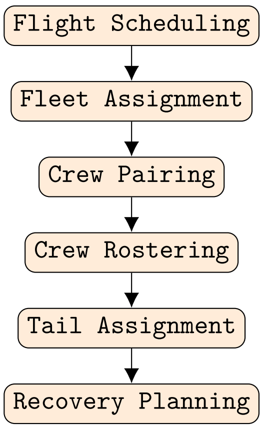

Figure 1.1: Sequential airline scheduling process

all scheduling problem is not solved optimally. Therefore more integrated solutions have also been considered, where two or more subproblems can be solved iteratively, considering some aspects of other subproblems, or a completely new integrated model of multiple problems is proposed. Although there exists no proposal to integrate all subproblems into a single problem, probably due to the complexity and size such a problem would have. Examples of integrated problems are Crew Scheduling with Tail Assignment and

Fleet Assignment with Flight Scheduling. There are some drawbacks when considering integrated optimization models since the subproblems are complex and large, even when considered individually. Integrating two or more subproblems can be very difficult to model, and the integrated optimization models will obviously become even larger in the number of required variables and constraints. The integrated optimization problems can therefore be more cumbersome to find good solutions to and fail in practice to perform better than the sequential approach in Fig. 1.1.

The goal of Flight Scheduling [6]–[15] is to construct a flight schedule, which is a list of flight legs specified by their arrival and departure airport, dates or frequency, and time. This stage is typically completed six months in advance. The considerations an airline usually has in Flight Scheduling are demand, ticket price, market share, airport slots, and non-stop flights versus connecting flights for certain origins and destinations. Since the sold flight tickets constitute the airline's revenue, the objective becomes to maximize the expected revenue.

Using the output of Flight Scheduling, i.e., the flight schedule, it is possible to consider Fleet Assignment. In Fleet Assignment [16]–[33], the goal is to cover the flight schedule with the existing fleet and maximize the profit. This means that we need to assign a fleet type to each flight leg while not exceeding the number of aircraft in the fleet. To maximize the profit, the goal is to match the demand to the capacity of the aircraft, as this minimizes the spill cost of lost passengers and operating costs.

Once the flight schedule and the fleet type are known for each flight, we can consider the Crew Scheduling problem. The Crew Scheduling problem [34]–[37] is typically partitioned into two problems, Crew Pairing and Crew Rostering. The reasons for separating the problems are the sheer size and complexity and the constraints and goals that differ. In Crew Pairing, the goal is to find anonymous legal pairings in the most cost-effective way such that the flight schedule is covered and the working contractual rules for the anonymous crew are respected. The cost is typically measured in the working time, and a legal pairing is a sequence of duties that consists of flight legs, typically representing one day of work, with rest and layovers in between. It starts and ends in a crew base and adheres to a set of additional rules.

Crew Rostering, on the other hand, focuses on finding monthly personalized rosters for each crew member such that all flights are covered, and the weekly and monthly rules for individual crew members are respected. The personalized rosters for each crew consist of pairings generated in the Crew Pairing phase and personal activities such as vacation, reserves, and training with time off in between. The typical objective is to maximize the roster qualities by fairness and/or requests from individual crew members. Crew Pairing is, therefore, more important than Crew Rostering when it comes to increasing profit.

Once there is a Fleet Assignment and/or Crew Scheduling solution in place, individual tails, the numbers that identify aircraft, must be assigned to each flight in the flight schedule while performing the required maintenance checks. This problem is an Aircraft Assignment problem, and specifically here the Tail Assignment problem [38]–[42]. The solution of Tail Assignment is a set of maintenance feasible aircraft routes, that is, sequences of flight legs, assigned to minimize the assignment cost.

Once all these problems are solved, the airline has successfully scheduled its flights, aircraft, and crew to maximize profit while respecting all rules. However, there are many uncertainties, as noted previously. As a result, schedules can be disrupted by unforeseen circumstances. To manage this new situation, it is often necessary to reevaluate the schedules, which is handled in the Recovery Planning phase [43], [44].

Ch. 2 and 3 will explain further the mathematical models and solution techniques commonly used.

1.2 Quantum Computing

Quantum mechanics [45] was discovered and developed during the first quarter of the 1900s, introducing concepts such as quantum superposition, quantum entanglement, and quantum measurement. The theory allowed for a greater understanding of the universe throughout the century, as it correctly described nature in regimes where classical mechanics failed.

The theory of computational complexity [46]–[48] and the performance of the classical computer progressed at great speed during the latter half of the 1900s, following Moore's law [49], [50] which says that the power of classical computers will double every two years. However, this law is currently breaking down as the hardware components become so small that quantum mechanical effects disturb their functionality. One possible way of furthering our computing capability is to propose a new model of computation. In 1979-1981 Benioff [51], Manin [52], and Feynman [53] independently proposed such concepts, namely a model of computation based on quantum mechanics. Feynman argued that to simulate quantum mechanical systems, such a model of computation might be required, as it seems to be intractable for a classical computer to do exactly without exponential resources and time. The idea was thus that a machine where information is embedded in quantum mechanical systems might be more powerful than a machine where the information is embedded in classical mechanical systems. Today we view quantum superposition and entanglement as resources of this model that are distinctly different from the resources of classical computation. Quantum entanglement allows for non-local operations compared to classical computing and quantum superposition allows the device to be in a superposition of all classical states. Indeed, quantum computers stand today as a possible model of computation that may violate the extended Church-Turing thesis, which says that a probabilistic Turing machine can simulate any reasonable model of computation in polynomial time. The principles of quantum computing can be found in Ref. [54] but will also be introduced in Ch. 4.

Following the proposals by Benioff, Manin, and Feynman, much insight has been obtained about controlling single quantum systems such as ion traps and superconducting qubits, quantum information, quantum algorithms, and how to construct a quantum computer [54], [55]. Deutsch showed in 1985 that universal quantum computing [56] is possible to realize in theory, along with a problem that can be solved in constant time by a quantum algorithm compared to linear time by a deterministic classical algorithm [57], albeit a probabilistic classical algorithm also solves the problem in constant time. Bernstein and Vazirani [58] were the first to show separation between quantum computation and classical computation, as they gave a problem that a quantum algorithm solves in constant time whereas linear time is required for both the deterministic and probabilistic classical algorithm. They also proposed a version of the quantum Fourier transform, which gives an exponential speedup compared to classical algorithms. Exponential speedup was also obtained by Simon's algorithm [59] shortly after, but what is now considered a defining major breakthrough in the field was discovered by Shor [60], who presented an algorithm, that uses the quantum Fourier transform as a building block, that solves the discrete log problem and in extension the integer factorization problem. This quantum algorithm provides exponential speedup over the best known classical algorithm, and is applicable to a problem we encounter on a regular basis, namely the RSA encryption [61], which is broken by Shor's algorithm. Moreover, Grover presented an unstructured database search algorithm with quadratic speedup [62], which has later also been used to propose several algorithms to solve discrete optimization problems. However, these algorithms rely on quantum error correction and hence fault-tolerant quantum computers, which have yet to be shown experimentally viable. Furthermore, some algorithms require specific oracle access, which is nontrivial to achieve experimentally.

What has been proposed instead is to consider hybrid quantum-classical variational algorithms such as the Quantum Approximate Optimization Algorithm (QAOA) [63] and the Variational Quantum Eigensolver (VQE) [64] in the so-called Noisy Intermediate-Scale Quantum [65] (NISQ) era of quantum computers. We will discuss the quantum variational algorithms designed to solve optimization problems further in Ch. 4.

Moreover, the idea of quantum computing has given rise to quantum complexity classes. The complexity class of greatest interest to us is Bounded-error Quantum Polynomial-time (BQP) [54], which is the quantum analog of Bounded-error Probabilistic Polynomial-time (BPP) [47]. Finally, we note that it has been shown that BPP$\subseteq$ BQP$\subseteq$PSPACE [58], but it is not known if there is a separation BPP$\neq$BQP, meaning that it is still not proven that quantum computers are more powerful than classical computers. This can be considered counterintuitive, as we just have mentioned quantum speedup. However, for the problems and quantum algorithms presented, the speedup is either given when we assume oracle access, which can't separate the classes, or when we do not know the hardness of the problem as in the case of integer factorization.

1.3 Contribution

I assisted with the following work related to the papers appended and discussed in this thesis:

- Paper A: I derived problem instances of interest, performed all numerical simulations, and was the main author of the manuscript

- Paper B: I derived the problem instances and assisted with reviewing the manuscript
- Paper C: I assisted in reviewing the manuscript and assisted in some numerical circuit simulations
- Paper D: I presented the instances to consider and assisted in reviewing the manuscript

1.4 Thesis Outline

In Ch. 2, the mathematical background is given for Multi-Commodity Network Flow problems and solution methodologies. A further explanation of the mathematical models used for the Aircraft Assignment problem Tail Assignment is then given in Ch. 3. Ch. 4 gives a review of the model of quantum computation and some quantum algorithms designed to solve integer programming problems. The appended papers are summarized in Ch. 5, and finally, we give our conclusions and suggestions for future research possibilities in Ch. 6.

CHAPTER 2

Network Flows and Mathematical Optimization

This chapter introduces the general minimum cost Multi-Commodity Network Flow Problem (MCNFP) as it models many airline planning problems, sometimes with additional constraints. We also summarize solution methodologies and focus on Dantzig-Wolfe decomposition, Column Generation, and in the integrality case Branch-and-Price.

2.1 The Multi-Commodity Network Flow Problem

Surveyed in [66] and [67], the MCNFP [68] model optimization problems in areas such as logistics, transportation, and telecommunication. The problem consists of a directed graph $G = (V, A)$ with a set of nodes V and arcs A. In addition, we have a set of K commodities that, in essence, differentiates the problem from the Single-Commodity Network Flow Problem (SCNFP). The goal is to ship B^k units of each commodity $k \in K$ across the graph from source nodes s_k to sink nodes t_k such that the sum of arc costs c_{ij}^k is as small as possible, whilst respecting capacity constraints on each arc u_{ij} and arc-flow variable x_{ij}^k.

Thus, similarly to the SCNFP, we wish to move commodities from the source to the sink subject to mass balance constraints, capacity constraints and a cost minimization. However, here the capacity constraints link all commodities together, which causes the MCNFP to be a more difficult problem to solve than the SCNFP. In particular, if we have integral variables as the Linear Programming (LP) relaxation [69] of the SCNFP, in this case, has integer solutions, whereas the MCNFP does not. Clearly, if the linking constraints are ignored, the MCNFP decomposes to $|K|$ SCNFPs that can be solved separately. There are two main formulations of these problems, an arc-flow formulation, and a path-flow formulation. We will now discuss these.

Arc-flow Formulation

The arc-flow formulation of the minimum cost MCNFP is the following

$$
\begin{aligned}
z^* = \text{minimize} \quad & \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k, && && (2.1)\\
\text{subject to} \quad & \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k, && \forall i \in V,\ \forall k \in K, && (2.2)\\
& \sum_{k \in K} x_{ij}^k \le u_{ij}, && \forall (i,j) \in A, && (2.3)\\
& x_{ij}^k \ge 0, && \forall (i,j) \in A,\ \forall k \in K, && (2.4)
\end{aligned}
$$

where

$$
b_i^k = \begin{cases} B^k & \text{if } i = s_k \\ -B^k & \text{if } i = t_k \\ 0 & \text{otherwise} \end{cases} . \qquad (2.5)
$$

The decision variables x_{ij}^k represent the flow of commodity k on each arc $(i,j) \in A$ in the directed graph. Eq. (2.1) is the linear cost function with cost coefficients c_{ij}^k associated with sending commodity k across arc (i,j). There are $|V||K|$ mass balance constraints, sometimes referred to as continuity constraints, in Eq. (2.2). Here the supply, demand, and continuity are secured by the coefficients b_i^k for each node and commodity. Each commodity has its own source node s_k and sink node t_k, and there can in general be several source and/or sink nodes, but here we only consider the case when we have a single source node and a single sink node for each commodity. The linking constraints of the commodities are given by the upper bound on the arc capacity u_{ij} in Eq. (2.3). Clearly these $|A|$ constraints are forcing us to solve the problem in this form, whereas if these constraints are ignored, it is possible to decompose the problem into $|K|$ separate problems.

Path-flow Formulation

We can reformulate the arc-flow formulation as a path-flow formulation. Since there are for each commodity k a set of P^k possible simple directed paths from s_k to t_k, we can associate the flow on each path with a decision variable f_p. We can then relate the arc-flow decision variable to the path-flow decision variable accordingly

$$x_{ij}^k = \sum_{p \in P^k} \delta_{ij}(p) f_p, \tag{2.6}$$

where $\delta_{ij}(p)$ is 0 if arc (i,j) is not in path p and 1 otherwise. We can also express the cost of a path p for commodity k as $c_p^k = \sum_{(i,j)\in A} \delta_{ij}(p) c_{ij}^k$. Regarding the mass balance constraints, we can remove some, as we require that the sum of all directed paths from source to sink for each commodity to deliver B^k units. Lastly, the capacity constraints are easily expressed in path-flow variables by replacing the arc-flow variables. Since we require that each arc-flow variable is non-negative, we also require this for the path-flow variables. The path-flow formulation is thus

$$\begin{aligned}
z^* = \text{minimize} \quad & \sum_{k\in K} \sum_{p\in P^k} c_p^k f_p, && && (2.7)\\
\text{subject to} \quad & \sum_{p\in P^k} f_p = B^k, && \forall k \in K, && (2.8)\\
& \sum_{k\in K} \sum_{p\in P^k} \delta_{ij}(p) f_p \leq u_{ij}, && \forall (i,j) \in A, && (2.9)\\
& f_p \geq 0, && \forall p \in P^k,\ \forall k \in K. && (2.10)
\end{aligned}$$

We now have an equivalent formulation of the problem where we have reduced $|V||K| + |A|$ constraints in the arc-flow formulation to $|K| + |A|$ constraints, but we have increased the number of variables from $|A||K|$ to $\sum_{k\in K} |P^k|$ which is in general exponential in the size of the directed graph.

2.2 Solution Approaches

The solution approaches to MCNFP [68, Chapter 17] are typically classified into three categories: price-directive decomposition, resource-directive decomposition, and partitioning methods.

The price-directive decomposition decomposes the problem into a main problem and $|K|$ subproblems where prices are put on the linking constraints. The mass balance constraints and individual arc flow constraints define the subproblems for each commodity along with an objective function to be minimized. The objective function is the original arc cost for a path with additional prices added. The role of the subproblems is to find improving paths for the main problem. The main problem sets the prices and connects the individual subproblems via the linking constraints. This is often done via Lagrangian decomposition, Dantzig-Wolfe decomposition and/or Column Generation. The two latter approaches will be presented in the following sections.

The idea of resource-directive decomposition is that we view the problem as a capacity allocation problem. We then separate the MCNFP into a resource allocation problem and $|K|$ additional minimum cost flow problems that depend on a fixed resource vector $\mathbf{r}$. This decomposition changes the arc-flow formulation to the following two problems

$$z^* = \text{minimize} \quad \sum_{k \in K} z_k(\mathbf{r}), \tag{2.11}$$

$$\text{subject to} \quad \sum_{k \in K} r_{ij}^k \leq u_{ij}, \qquad \forall (i,j) \in A, \tag{2.12}$$

$$r_{ij}^k \geq 0, \qquad \forall (i,j) \in A,\ \forall k \in K, \tag{2.13}$$

and

$$z_k(\mathbf{r}) = \text{minimize} \quad \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k, \tag{2.14}$$

$$\text{subject to} \quad \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k, \qquad \forall i \in V, \tag{2.15}$$

$$0 \leq x_{ij}^k \leq r_{ij}^k, \qquad \forall (i,j) \in A. \tag{2.16}$$

These problems are solved iteratively, where we reallocate capacities such that the solution is improved in each iteration.

For partitioning methods, one uses the fact that the MCNFP is an LP with embedded single-commodity network flow problems. One useful method that has been developed for SCNFP is that spanning tree solutions are basic feasible solutions in the simplex algorithm [70]. By generating improving spanning trees, the SCNFP can be solved. This idea is expanded upon for MCNFP, where additional arcs are required to ensure that the linking constraints are satisfied.

To summarize, all three categories of solution approaches to the MCNFP share the fact that we separate the problem into several problems, that each only considers one commodity. In this thesis, we focus on a price-directive method, namely the Dantzig-Wolfe decomposition and Column Generation. Our motivation for this choice is that we consider the current state-of-the-art implementation to solve problems of interest in the airline planning process.

Dantzig-Wolfe Decomposition

The Dantzig-Wolfe decomposition was first presented in [71], and constitutes a method for decomposing a linear program such that we obtain a formulation based on extreme rays and points in domains that are defined by constraints in the original formulation. If we consider the linear program

$$\begin{aligned}
\text{minimize} \quad & \mathbf{c}_1^T\mathbf{x}_1 + \mathbf{c}_2^T\mathbf{x}_2 + \cdots + \mathbf{c}_N^T\mathbf{x}_N, && (2.17)\\
\text{subject to} \quad & D_1\mathbf{x}_1 + D_2\mathbf{x}_2 + \cdots + D_N\mathbf{x}_N \leq \mathbf{d}, && (2.18)\\
& \begin{bmatrix} A_1\mathbf{x}_1 & & & \\ & A_2\mathbf{x}_2 & & \\ & & \ddots & \\ & & & A_N\mathbf{x}_N \end{bmatrix} \leq \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}, && (2.19)\\
& \mathbf{x}_1,\ \mathbf{x}_2, \ldots, \mathbf{x}_N \geq \mathbf{0}, && (2.20)
\end{aligned}$$

where all variables are linked by the D_i matrices, and the A_i matrices are separate constraints for each vector $\mathbf{x}_i$.

Assume for a moment that we want to formulate a problem with only the constraints in Eq. (2.18) involving the linking constraints. This means that we somehow wish to remove the rest of the constraints. Let us begin by defining the sets

$$X_i = \{\mathbf{x}_i \geq 0 \mid A_i\mathbf{x}_i \leq \mathbf{b}_i\} \quad \forall i = 1, \ldots, N. \qquad (2.21)$$

If these sets are convex and nonempty, a point $\mathbf{x}_i \in X_i$ can be written as a convex combination of the extreme points $\bar{\mathbf{x}}_i^p \in P_i$ and a conical combination of extreme rays $\tilde{\mathbf{x}}_i^r \in R_i$ [72, Chapter 3]

$$\mathbf{x}_i = \sum_{p \in P_i} \lambda_i^p \bar{\mathbf{x}}_i^p + \sum_{r \in R_i} \lambda_i^r \tilde{\mathbf{x}}_i^r, \quad \sum_{p \in P_i} \lambda_i^p = 1, \quad \lambda_i^l \geq 0,\ l \in P_i \cup R_i. \tag{2.22}$$

It is then possible to express the original linear program by substituting in Eq. (2.22), which gives us the following equivalent formulation

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \Big(\sum_{p \in P_i} \lambda_i^p \mathbf{c}_i^T \bar{\mathbf{x}}_i^p + \sum_{r \in R_i} \lambda_i^r \mathbf{c}_i^T \tilde{\mathbf{x}}_i^r \Big), & (2.23) \\ \text{subject to} \quad & \sum_{i=1}^{N} D_i \Big(\sum_{p \in P_i} \lambda_i^p \bar{\mathbf{x}}_i^p + \sum_{r \in R_i} \lambda_i^r \tilde{\mathbf{x}}_i^r \Big) \leq \mathbf{d} \quad \mid \vec{\pi}, & (2.24) \\ & \sum_{p \in P_i} \lambda_i^p = 1, \quad i = 1, \ldots, N \quad \mid q_i, & (2.25) \\ & \lambda_i^l \geq 0, \quad \forall l \in P_i \cup R_i, \quad i = 1, \ldots, N. & (2.26) \end{aligned}$$

We have at this point successfully removed the constraints we desired in our formulation by considering the extreme points and rays in the polyhedrons X_i. The reformulated problem now has decision variables λ_i^l, and we will denote this problem as the Master Problem (MP).

Column Generation

If it is possible to find all extreme points and rays and the number of extreme points and rays is not too large, the problem obtained from the Dantzig-Wolfe decomposition can be solved directly. However, this is not the case in general. Rather, the number of extreme points and rays can be very large, even exponentially large in the input size. We, therefore, seek a method where we only consider some subset of all variables. This is where the Column Generation algorithm [73] becomes essential.

The problem with only a subset of variables $R_i' \subset R_i$ and $P_i' \subset P_i$ for $i = 1, \ldots, N$ is called the Restricted Master Problem (RMP). By solving the RMP, the optimal primal and dual variables can be obtained. Additionally, the reduced cost of variables λ_i^p and λ_i^r for given dual variables $\vec{\pi}$ and q_i are

$$\bar{c}_i^p = \mathbf{c}_i^T \bar{\mathbf{x}}_i^p - (D_i \bar{\mathbf{x}}_i^p)^T \vec{\pi} - q_i \quad \text{and} \qquad \bar{c}_i^r = \mathbf{c}_i^T \tilde{\mathbf{x}}_i^r - (D_i \tilde{\mathbf{x}}_i^r)^T \vec{\pi}, \tag{2.27}$$

respectively. Since a variable with negative reduced cost can improve the solution of the RMP, we wish to find the smallest by solving the Pricing Problem (PP)

$$\text{minimize}_{\mathbf{x}_i \in X_i} (\mathbf{c}_i - D_i^T \vec{\pi})^T \mathbf{x}_i - q_i, \tag{2.28}$$

which is equivalent to solving the following two minimization problems

$$\min\Big(\text{minimize}_{p \in P_i} (\mathbf{c}_i - D_i^T \vec{\pi})^T \bar{\mathbf{x}}_i^p - q_i,\ \text{minimize}_{r \in R_i} (\mathbf{c}_i - D_i^T \vec{\pi})^T \tilde{\mathbf{x}}_i^r\Big).$$

By solving the problems, we can find a column that can enter the basis (i.e., the RMP) which will either be an extreme point

$$\begin{pmatrix} \mathbf{c}_i^T \bar{\mathbf{x}}_i^p \\ D_i \bar{\mathbf{x}}_i^p \\ 1 \end{pmatrix}$$

or an extreme ray

$$\begin{pmatrix} \mathbf{c}_i^T \tilde{\mathbf{x}}_i^r \\ D_i \tilde{\mathbf{x}}_i^r \\ 0 \end{pmatrix}.$$

When there are no variables $p \in P_i$ or $r \in R_i$ in any pricing problem $i = 1, \dots, N$ with a negative reduced cost, the optimal RMP has been found, and also the optimal solution to the original problem presented in Eq. (2.17)-(2.20).

The Column Generation algorithm viewed from a Dantzig-Wolfe decomposition perspective is explained as iteratively solving the RMP and the $2 \cdot N$ pricing problems based on the dual variables. If we then solve each pricing problem, and we find new extreme points or rays with negative reduced cost, we introduce these variables in the basis (the RMP). Repeating this process will generate an improved solution in each iteration. When no extreme point or ray with a negative reduced cost is found, the problem is solved optimally. However, we are not necessarily restricted to using Dantzig-Wolfe decomposition to employ the Column Generation algorithm for a problem, and it is possible to use it on the MCNFP directly where we have $|K|$ pricing problems with the form

$$z_k^* = \text{minimize} \quad \sum_{(i,j)\in A} (c_{ij}^k - \pi_{ij})x_{ij}^k - \sigma^k, \tag{2.29}$$

$$\text{subject to} \quad \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k = b_i^k, \qquad \forall i \in V, \tag{2.30}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \tag{2.31}$$

and the Master Problem in the path-flow formulation of the MCNFP.

Branch-and-Bound and Branch-and-Price

Given that we have an MCNFP where the variables are continuous, linear programming based techniques work well. However, it is often the case that we have integer valued variables. In this case, we do not accept fractional solutions nor is the domain convex.

One standard method used to find integer solutions is Branch-and-Bound [74], [75]. Consider here that we are applying Branch-and-Bound to an integer linear program

$$\text{ILP} = \min \left\{ \sum_{i=1}^{n} c_i x_i : \vec{x} \in S \right\}$$

where $S = \left\{ \vec{x} \in \mathbb{Z}_+^n : \sum_{i=1}^n a_{ji}x_i \geq b_j \ \forall j = 1, \ldots, m \right\}$. First, we consider what happens if the variables are relaxed to be continuous. We know from linear programming theory that we will obtain either a lower bound, an integer solution, or that the problem is infeasible.

If we find either an integer solution or an infeasible solution, we have either obtained the optimal solution or found that there is no integral solution, and we can stop. However, if we obtain some fractional solution $\mathbf{x}^0$, we have obviously not reached our goal. In Branch-and-Bound we, therefore, choose to create $k \geq 2$ subproblems, where we choose the subproblems to be in disjoint domains $S_1, \ldots, S_k$ that exclude the solution $\mathbf{x}^0$.

One way of creating disjoint domains with respect to the fractional variables in solution $\mathbf{x}^0$ is to add constraints for fractional variables in our solution $x_j \leq \lfloor x_j^0 \rfloor$ in one of the subproblems and $x_j \geq \lceil x_j^0 \rceil$ in another. The original problem and subproblems can be visualized as a tree, where the nodes represent the optimization problems, with one parent and two or more children.

Since we now have two or more new problems to explore, we again meet a similar situation as for our relaxed ILP, but for the newly created subproblems. This means that if a subproblem is found to be infeasible, the node is pruned, i.e., the node representing one of the subproblems is not explored further as we do not create any more subproblems. Instead, we choose a new subproblem to explore that has been generated previously, and we say that the node is *pruned by infeasibility*. If we find that a solution to a subproblem is integral, we also do not create any further subproblems and prune this node. As we have found an optimal partial solution, a candidate incumbent z_i, for this specific region and instead choose a new subproblem to explore. Imagine now that we also have access to some other feasible solution to the ILP, which we denote z^* and call the incumbent. If we find that a subproblem yields a fractional solution greater or equal to z^*, this region cannot contain any integer solutions that would improve upon the one we already have, and there is no point in creating more subproblems to the node we are currently exploring. This particular node is thus *pruned by bound*, and we choose a new subproblem to explore that has been generated previously. On the other hand, if the fractional solution is smaller than the incumbent, we create new subproblems, before we pick a new node to explore.

With these ingredients, we can state the Branch-and-Bound algorithm. The algorithm begins by first initializing a list $\mathcal{L}$ with the node N_0 that represents our relaxed ILP. We also initialize an incumbent $z^* = \infty$. We then pick a node in our list and attempt to solve this problem. If there are no feasible solutions, we pick another node from our list as this node is pruned by infeasibility. If we obtain an integral solution the node is also pruned, now by integrality, and if the integral solution is less than our incumbent, we update it. We then pick a new subproblem from the list. If we obtain a fractional solution worse than our incumbent, we prune this node by bound. Finally, if a subproblem can not be pruned by infeasibility, bound or integrality, the subproblem is partitioned into $k \geq 2$ nodes representing k subproblems, which are children to the current subproblem we are exploring in the tree. The k subproblems are then added to a list $\mathcal{L}$ of unexplored subproblems and a new subproblem is chosen to be explored. When there are no unexplored subproblems left, the algorithm terminates and returns the incumbent solution and the corresponding assignment.

We remark that the Branch-and-Bound algorithm is exponential in the worst case. However, exhaustive search is avoided by pruning nodes of the tree, giving more acceptable running times in practice. Finally, when we solve each node with the Column Generation method, the algorithm is called Branch-and-Price [76].

CHAPTER 3

Airline Scheduling Models

Mathematical optimization models for airline scheduling problems can differ significantly depending on the type of network, planning horizon, network representation, uncertainties, objective function, and considered constraints. Indeed, it does not seem surprising that depending on the assumptions made for a problem at hand in the airline planning process that we end up with very different mathematical models for something we categorize as approximately the same problem. Furthermore, a so-called good model depends very much on how well the model represents reality and how fast a sufficiently good solution can be obtained. Therefore, we must consider at least the trade-off between the time spent to find a solution versus the solution quality, and we must be aware that the models typically vary for different airlines, and can change if the industry changes.

Now, since the papers discussed in Ch. 5 have only considered a variant of Aircraft Assignment called Tail Assignment, we will restrict the coming section to Aircraft Assignment and elaborate on Tail Assignment specifically. Even though we disregard the details of the other problems in the airline planning process, we note that the problems Fleet Assignment, Crew Pairing, and Crew Rostering have, in many cases, in common with Aircraft Assignment the property that they can be modeled as networks, which we must send either crew or aircraft through, and that the problems are large. Therefore, the model types and solution approaches for the problems are similar.

3.1 Aircraft Assignment and its Variations

As mentioned previously, many aspects of Aircraft Assignment can vary, leading to several names being used, both in industry and academia. Four common names are *Aircraft Routing* [77]–[81], *Aircraft Maintenance Assignment* [79], [82]–[92], *Through Assignment* [93]–[97] and *Tail Assignment* [38], [42], [98]. The name *Aircraft Assignment* is the common denominator, in the sense that Aircraft Routing, Through Assignment, Aircraft Maintenance Assignment, and Tail Assignment can be considered to be an Aircraft Assignment problem since we wish to assign flights, i.e., routes, to aircraft. In contrast, a general Aircraft Assignment model does not necessarily capture all the modeling aspects of Tail Assignment.

Having different modeling considerations leads of course to the fact that the mathematical models differ, to varying degrees. The models can differ in the ultimate goal, as some airlines consider Aircraft Assignment to be a feasibility problem, whereas others consider it to be an optimization problem. Here, robustness, which refers to how sensitive the solution is to disruptions, might also be more or less important to consider. A typical case of feasibility is Aircraft Maintenance Assignment, where it is often only required to find maintenance feasible routes assigned to aircraft while disregarding the cost of the assignment. Through Assignment, on the other hand, ignores many of the maintenance constraints, and the goal is instead to maximize the through values. The through values are defined as the desirability of one-stop services, i.e., multi-leg flights without aircraft changes. By maximizing this quantity, we minimize the number of aircraft changes for desirable connections, which allows the airline to raise the ticket price and increase the profit. Aircraft Routing refers to when we consider both maintenance requirements and through values.

Often, these three problems are considered earlier in the planning process than Tail Assignment as we do not always expect to obtain an executable solution without some manual changes, whereas Tail Assignment focuses on being able to produce a solution that can be executed without extra manipulation. For example, it can be desirable to solve Aircraft Maintenance Assignment directly after Fleet Assignment to show that it is possible to find at least some assignment that respects a subset of maintenance rules before considering crew scheduling.

Regardless of what variant of Aircraft Assignment we consider, one further example of diverging models is when we consider cyclic problems, which are problems where the flight schedule is approximately repeated each week or day. Here, it can be sufficient to require the aircraft to land at maintenance stations at the end of each day. Thus complying with some maintenance constraints without explicitly having them in the model, but implicitly enforcing them via the network structure [79]. With this assumption, once a solution is obtained, the schedule can be repeated to get a solution with a longer time horizon. The solutions are then often modified for dated problems, i.e., problems with distinct start and end dates, to consider deviations and improve some feasibility and/or optimality issues. If, on the other hand, the flight schedule has no such regularity, solving cyclic problems is not very useful, and we instead consider only the dated problems. Here it is usually impossible to model all maintenance requirements via the network structure, and we need to have explicit constraints in our model[1].

Another consideration is flexibility, meaning that the model can be adjusted according to what priorities an airline have depending on when they solve it, i.e., if we are several months, weeks, or days away from day-of-operation.

To reiterate, we generally wish to assign all flights to aircraft. Since this assignment corresponds to aircraft routes, we also want these routes to be, at the very least, maintenance feasible in some sense[2]. The maintenance requirements are given by aviation authorities[3], aircraft manufacturers, and airlines, which typically provide stricter constraints compared to the former two sources of constraints.

Tail Assignment

In Tail Assignment, the goal is to assign each flight exactly once to aircraft such that the operational cost is minimized and all operational constraints

[1]Maintenance are often modeled as restricted resources in the network.

[2]This means that if Through Assignment is solved, additional changes are most likely required to comply with maintenance constraints.

[3]Federal Aviation Administration in case of the USA.

such as maintenance, preassigned activities, and prohibited activities are respected. One notable difference is that this problem considers tail (aircraft) specific constraints, which models such as Aircraft Routing often do not consider. Furthermore, even though we have spoken about flight legs so far, Tail Assignment plan in its model something we call activities. Activities can be flight legs, sequences of flight legs, maintenance, and other ground activities. To note, one strength of Tail Assignment is that the model can capture aspects ranging from Aircraft Routing to Aircraft Maintenance Assignment, Through Assignment, Fleet Assignment, and Recovery Planning. Although Aircraft Assignment is typically separated for each fleet type, Tail Assignment is more general and allows multiple fleet types which can be required for the final feasibility of the solution. To model all requirements, Tail Assignment only solves dated problems but has, in principle, no limit on the time horizon. This means that Tail Assignment does not capture cyclic problems, which can be considered a weakness of the model. However, in terms of flexibility, scope, and preciseness in the sense that the solution should be executable without extra manipulation, the model is very well-suited for real-world problems.

In [38] Tail Assignment is presented for two different formulations, one which is path-flow based and exponentially large in the number of variables, or rather feasible routes, and linear in the number of constraints, i.e., the number of constraints are the number of flight legs given as input[4]. The second model is arc-flow based where the number of variables is quadratic in the number of activities and linear in the number of aircraft but has more constraints, and the constraints are viewed as complex. Furthermore, although the number of variables is polynomial in the size of the input, the model becomes for practical problems very large. The two models are related via methods described in Ch. 2 as the latter model is classified as a resource-constrained integer minimum cost MCNFP and given explicitly in Eq. (3.1)-(3.7). We can consider the model below to be represented by a directed graph where each node represents a connection between activity i and j for each aircraft. In the network, we use five different types of sets to model our problem. T is the set of aircraft, F is the set of flight legs, P_t is the set of preassigned activities for tail t, R_t are the forbidden activities for tail t and M is the set of maintenance activities. The 0-1 decision variables in Eq. (3.7) represent which activities and thus connections the aircraft should cover where

[4]Here we are disregarding some vertical constraints compared to the model in practice.

each decision variable x_{ijt} is associated to a cost c_{ijt}, giving us the objective function in Eq. (3.1) to minimize. Constraint in Eq. (3.2) is a continuity constraint that ensures that a path is associated with an aircraft, albeit the requirement on the source and sink for each aircraft needs to be added such that exactly one aircraft is sent on each path. The covering constraint is given in Eq. (3.3) and ensures that all flight legs are covered exactly once. Then, the requirement for preassigned activities and forbidden activities for each aircraft are given in Eq. (3.4)-(3.5).

$$\begin{aligned}
\text{minimize} \quad & \sum_{i\in F}\sum_{j\in F}\sum_{t\in T} c_{ijt}x_{ijt}, && && (3.1)\\
\text{subject to} \quad & \sum_{j\in F} x_{jit} - \sum_{j\in F} x_{ijt} = 0, && \forall i \in F,\ \forall t \in T, && (3.2)\\
& \sum_{j\in F}\sum_{t\in T} x_{ijt} = 1, && \forall i \in F, && (3.3)\\
& \sum_{j\in F} x_{ijt} = 1, && \forall i \in P_t,\ \forall t \in T, && (3.4)\\
& \sum_{j\in F} x_{ijt} = 0, && \forall i \in R_t,\ \forall t \in T, && (3.5)\\
& r_{im} \le l_m, && \forall i \in F,\ \forall m \in M, && (3.6)\\
& x_{ijt} \in \{0,1\}, && \forall i \in F,\ \forall j \in F,\ \forall t \in T. && (3.7)
\end{aligned}$$

The most complex constraints for the model are, unexpectedly, associated with the maintenance in Eq. (3.6). These are defined recursively for each variable $x_{i'it} = 1$ where

$$r_{im} = \begin{cases} r_m^t & \text{if } i \text{ is carry-in activity for aircraft } t \\ s_{im} & \text{if maintenance } m \text{ possible between activities } i' \text{ and } i \\ s_{im} + r_{i'm} & \text{if maintenance } m \text{ not possible between activities } i' \text{ and } i \end{cases} .$$

Here, s_{im} is the resource consumption of maintenance m for activity i, r_m^t is the initial maintenance consumption for maintenance task m and aircraft t, r_{im} is the total resource consumption up to activity i, and l_m is the upper bound on maintenance m.

Notably, in [98] the constraints are clarified such that

$$r_{jmt} = s_{jmt} + \sum_{i \in F} v_{ijmt} r_{imt} x_{ijt} \leq l_{mt}$$

where v_{ijmt} is 1 if maintenance m is not possible for aircraft t between flight leg (or activity) i and j. The consequence is that there is a constraint for each activity, maintenance, and aircraft which can lead to some simplifications. However, the resource maintenance consumption r_{jmt} is still defined recursively and remains complicated.

For both versions, it is possible to decompose the problem via, e.g., Dantzig-Wolfe decomposition discussed in Ch. 2. The decomposition method presents us the master problem, which is an LP relaxed Set Partitioning problem and $|T|$ pricing problems, where each pricing problem is a resource-constrained shortest path problem. Explicitly, the first model we mentioned for Tail Assignment is obtained by modifying the decision variables to path variables, i.e., route variables x_r, giving us the Set Partitioning model below

$$\begin{aligned}
\text{minimize} \quad & \sum_{r \in R} c_r x_r, & & (3.8) \\
\text{subject to} \quad & \sum_{r \in R} a_{fr} x_r = 1, \quad \forall f \in F, & & (3.9) \\
& x_r \in \{0,1\}, \qquad \forall r \in R. & & (3.10)
\end{aligned}$$

This model associates each route with a route cost c_r in Eq. (3.8) and ensures that each flight is covered exactly once in Eq. (3.9). The number of variables is in the worst case exponential in the size of flight legs, but since we require that all routes must be feasible according to constraints in Eq. (3.2), (3.4), (3.5), (3.6), and (3.7) this number is reduced. However, explicit enumeration is typically intractable still and a well-known issue for these types of problems.

Finally, we note here that to find feasible integer solutions, via standard methods such as Branch-and-Price, we are faced with two NP-hard optimization problems.

CHAPTER 4

The Model of Quantum Computation and Quantum Optimization

The previous chapters are aimed at giving a solid understanding of classical algorithms and modeling considerations typically considered in the airline industry. However, since this thesis is concerned with how quantum algorithms can be employed for such problems, we now present a foundation of quantum computing [54] in Sec. 4.1 and following that we present, in Sec. 4.2, the two quantum algorithms that have been explored in the appended papers.

4.1 Model of Quantum Computation

Two popular models of quantum computation are the quantum circuit model [99] and the quantum Turing machine [56], [58]. The models are equivalent since they can simulate each other in polynomial time [100]. Here we will present the quantum circuit model since it corresponds relatively straightforwardly with algorithms such as QAOA.

Quantum States and Qubits

The quantum mechanical systems we study in this thesis are of some finite dimension N. For such a quantum system, we can express its state as a column vector of l_2-norm 1 in a N dimensional complex linear space $\mathbb{C}^N$, i.e., a Hilbert space $\mathcal{H}$. We can fix an orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$, using the Dirac notation[1] and express a pure state[2] as a superposition of the basis states

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$$

where the l_2-norm requirement means that $\sum_{i=0}^{N-1} |\alpha_i|^2 = 1$. A quantum bit is the typical building block of a quantum circuit and is the quantum analog of the classical bit. The quantum bit is a two-dimensional quantum system with states in $\mathbb{C}^2$, the orthonormal basis is most commonly chosen to be

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

which means that a general state of a quantum bit is expressed as $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, where α_0 and α_1 are complex numbers. The vector space we consider has the inner product

$$\langle \phi | \psi \rangle = \sum_{i=0}^{N-1} \beta_i^* \alpha_i$$

where $\langle \phi|$ is the conjugate transpose of $|\phi\rangle = \sum_{i=0}^{N-1} \beta_i |i\rangle$, i.e., $\langle \phi| = (|\phi\rangle)^\dagger$.

Suppose now that we have two distinct quantum mechanical systems in Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ of dimensions N_1 and N_2 with orthonormal basis $\{|i\rangle_1\}_{i=0}^{N_1-1}$ and $\{|j\rangle_2\}_{j=0}^{N_2-1}$, respectively. We can describe the composite quantum system by the tensor product of the two Hilbert spaces $\mathcal{H}_1 \otimes \mathcal{H}_2$ and obtain an orthonormal basis via tensor products[3] accordingly

$$|ij\rangle_{12} = |i\rangle_1 \otimes |j\rangle_2\,,\ \forall i = 0, \ldots, N_1 - 1\ , \forall j = 0, \ldots, N_2 - 1.$$

[1] In the Dirac notation we call the column vector a ket vector and the row vector a bra vector.

[2] A mixed state is a probability distribution of pure states.

[3] Typically we abbreviate the tensor product $|i\rangle \otimes |j\rangle$ as $|i\rangle |j\rangle$ or $|ij\rangle$.

Clearly, the dimension of the composite system's Hilbert space is $N_1 \times N_2$. We can express a general quantum state for the composite system in exactly the same manner as for a single system in the basis of the composite system

$$|\phi\rangle_{12} = \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} \alpha_{ij} |i\rangle_1 |j\rangle_2 .$$

If we consider qubits again, in this case two qubits, we get the orthonormal basis $|0\rangle = |0\rangle \otimes |0\rangle$, $|1\rangle = |0\rangle \otimes |1\rangle$, $|2\rangle = |1\rangle \otimes |0\rangle$ and $|3\rangle = |1\rangle \otimes |1\rangle$.

In order to describe a composite system that consists of n quantum mechanical systems with dimensions $N_1, N_2, \ldots, N_n$ we need a Hilbert space of dimension $N_1 \times N_2 \times \cdots \times N_n$. In the case where we consider n qubits the Hilbert space is of dimension 2^n and the states are column vectors in $\mathbb{C}^{2^n}$ with orthonormal basis $\{|i\rangle\}_{i=0}^{2^n-1}$ where $|i\rangle$ is a column vector where the entries are zeros, except in position $i+1$ which has entry one.

Unitary Evolution of Quantum States

Unitary transformation is one of the basic operations a closed quantum system can undergo. This means that we describe the evolution in time of a closed quantum system with unitary operators, where a unitary operator U is such that $U^\dagger U = \mathbb{1}$ when we take its matrix representation[4]. Consequently, time evolution and any unitary transformation preserve the norm of quantum states, and evolution is reversible. The unitary time evolution operator relates a state $|\psi\rangle$ at time t_1 to the state $|\psi'\rangle$ at time t_2 as

$$|\psi'\rangle = U |\psi\rangle .$$

In Dirac notation, we can express the unitary operator via the outer product of the orthonormal basis $|i\rangle \langle j|$. The unitary operator then has the following matrix representation

$$U = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} u_{ij} |i\rangle \langle j| , \tag{4.1}$$

and each column $\mathbf{u}_i$ describes how the operator acts on basis state $|i\rangle$. In the case when time is continuous, the evolution is governed by the Schrödinger

[4]Here we use the convention of $\mathbb{1}$ being the identity matrix.

equation

$$i\hbar \frac{\partial |\psi\rangle}{\partial t} = H|\psi\rangle, \tag{4.2}$$

where H is the Hamiltonian of the quantum system and $\hbar$ is Planck's constant divided by 2π. We can relate the differential equation to the unitary operator easily when the Hamiltonian is time-independent, giving us

$$U = e^{-iH(t_2-t_1)/\hbar}. \tag{4.3}$$

Finally, we note that we can express any unitary operator as $U = e^{iA}$ for some Hermitian[5] operator A.

Quantum Gates

Quantum gates constitute the second building block element of quantum circuits and are unitary operators that evolve the quantum system in time. Common one qubit gates are the Pauli-gates

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \ \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \text{ and } \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

As well as the Hadamard-gate, phase-gate, and $\pi/8$-gate below

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \ S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \text{ and } T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}.$$

Common two-qubit gates are controlled-U gates that act on a control qubit and a target qubit

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix}.$$

This gate does not change the state if the control qubit is in the state $|0\rangle$, but if the control bit is in state $|1\rangle$ it applies a gate

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

[5]A hermitian matrix A is such that $A^\dagger = A$.

to the target qubit. The CNOT-gate is a controlled-U gate where U is the σ_x-gate. A universal quantum gate set is CNOT, Hadamard (H), phase (S) and $\pi/8$ (T) according to the Solovay-Kitaev theorem [54, Appendix 3], as it is possible to approximate any other unitary gate arbitrarily well.

Quantum Measurements

Quantum mechanics prohibit us to observe a quantum state $|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$ in the sense that we can determine all amplitudes α_i. What quantum mechanics allows for instead are measurements that constitute a third element required for a quantum computational model. If we measure the state $|\psi\rangle$ in its orthonormal basis[6] we will observe an outcome i with probability $|\alpha_i|^2$ and the system will be in the state $\frac{\alpha_i}{|\alpha_i|} |i\rangle$ after the measurement.

Measurement is the second elementary operation a quantum system can undergo and is not described by a unitary operator. Instead, we can describe a measurement in the computational basis, or any other basis, with projective operators. A projective measurement is described by an observable M, which is a Hermitian operator with eigenvalues $\lambda_m,\ m = 1, 2, \ldots, K$. The observable is related to projection operators $\{P_m\}_{m=1}^K$. Each projective operator P_m is a projection onto the eigenspace corresponding to eigenvalue λ_m. We write the observable with respect to the different outcomes m as the sum

$$M = \sum_{m=1}^{K} \lambda_m P_m \ s.t. \sum_{m=1}^{K} P_m = \mathbb{1} \text{ and } P_m P_{m'} = \delta_{mm'} P_m.$$

Assuming that the system was in state $|\psi\rangle$ prior to the measurement of the observable, we have a probability

$$P(m) = \langle\psi| P_m |\psi\rangle$$

that outcome m is observed and if outcome m is observed the state collapses to

$$|\psi'\rangle = \frac{P_m |\psi\rangle}{\sqrt{\langle\psi| P_m |\psi\rangle}}.$$

A measurement in the computational basis is a projective measurement and therefore given by the projection operators $P_i = |i\rangle \langle i|,\ \forall i = 0, 1, \ldots, N-1$.

[6]Usually referred to as the computational basis.

In the one qubit case, where we measure σ_z, this corresponds to

$$P_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad P_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

with eigenvalues $+1$ and -1, respectively.

Quantum Circuits

A classical circuit [47] is a finite directed acyclic graph $G = (E, V)$ with n input nodes that take the input bit values, m output nodes, and internal nodes that each is one of the gates AND, OR, and NOT. The edges of the graph, also called wires, each carry one bit. Each internal node performs logical operations on the bits. Here, we note that the gates AND and OR have fanin, the number of incoming edges, two and fanout, the number of outgoing edges, one. The NOT-gate, on the other hand, has fanin one and fanout one. We note that for classical circuits, we are permitted to copy a bit, which is not allowed in quantum circuits due to the no-cloning theorem. Such a circuit G implements a function $f : \{0,1\}^n \to \{0,1\}^m$, and is a boolean circuit in the case when there is a single output node. This is the classical circuit model.

The quantum circuit model, see an example of a quantum circuit in Fig. 4.1, is defined similarly to the classical circuit model. The classical bits are replaced by quantum bits, the edges of the graph carry qubits, and quantum gates replace the classical gates. Moreover, fanin must be the same as fanout. Finally, quantum measurements are required to be added in order to observe the outcome such that we obtain the classical output bits.

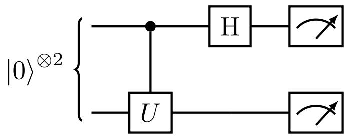

Figure 4.1: Quantum circuit with two input qubits in state $|00\rangle$ where we first apply a controlled-U gate, second the Hadamard gate, and finally both qubits are measured giving two classical bits as output

For the classical circuit model, we can define the notion of efficiency. We say that the size of the circuit is the number of nodes in the circuit, and the circuit is considered efficient if there exists a polynomial-sized circuit with respect to the input size that computes a function. Similarly, for the quantum circuit model, we use the same notion of efficiency, and we require that the size of the quantum circuit is polynomial in the input size, where each gate acts on at most three qubits[7].

4.2 Quantum Optimization Algorithms

As mentioned previously, various types of optimization problems frequently appear in industry and academia. Examples beyond logistics and transportation are the kidney swap problem in health care, social network optimization on graphs, and quantum transpilation. We, therefore, consider it valuable, both from a theoretical standpoint and in practice, to understand if quantum algorithms can improve upon classical algorithms designed to solve integer programs. However, since it was shown in [62], [101] that the best speedup we can expect for NP-complete problems is quadratic in the black box setting, we should perhaps consider approximate or heuristic quantum algorithms rather than exact algorithms.

Nonetheless, one can roughly categorize quantum optimization algorithms into two categories. The first category is nonheuristic algorithms. These algorithms have provable complexity behavior with respect to time and space and solution quality. In this category, we have Grover's algorithm and extensions where Grover's algorithm is embedded in a classical algorithm. The second category is heuristic algorithms, such as the Quantum Annealing Algorithm and the Quantum Approximate Optimization Algorithm, which we will discuss in the coming sections.

Adiabatic Quantum Computation and Quantum Annealing

Adiabatic quantum computation [102] is a universal paradigm of quantum computing proven to be as powerful as the circuit model discussed in Sec. 4.1 and is based on the adiabatic theorem [103]. The adiabatic theorem describes what happens to a quantum system, initialized in a non-degenerate eigenstate of an initial Hamiltonian H_0, that is changed continuously and adiabatically, or infinitely slowly, to a final Hamiltonian H_1. One example of such a situ-

[7]There exists a universal gate set that includes the Toffoli gate.

ation is when an external magnetic field is changed slowly for an interacting quantum spin system but has also been used to construct a whole separate paradigm of computation.

We can explicitly construct a time-dependent Hamiltonian that describes this situation as

$$H(t) = \frac{(T-t)}{T} H_0 + \frac{t}{T} H_1, \tag{4.4}$$

where T is a measure of how fast the system changes and governs the evolution time from H_0 to H_1. Assuming that the two Hamiltonians H_0 and H_1 act on an n qubit system, H_0 and H_1 do not commute, that the instantaneous eigenenergies $E_0(t) < E_1(t) < \cdots < E_{2^n-1}(t)$ of $H(t)$ are distinct for the evolution time and the system is initialized in the ground state $|e_0(0)\rangle$ of $H(0) = H_0$, we have the following bound on the time required to ensure that the evolution is adiabatic

$$\frac{\max_{s\in[0,1]} | \langle e_1(s)| \partial_s H(s) |e_0(s)\rangle |}{\min_{s\in[0,1]} |E_1(s) - E_0(s)|^2} \ll T,$$

where $s = t/T$. The bound above tells us that as long the bound is respected, the initial state $|e_0(s=0)\rangle$ has evolved to $|\psi\rangle = |e_0(s=1)\rangle$ after the evolution time $t = T$.

Now we direct our attention to the fact that the adiabatic theorem can be used to construct an algorithm for solving discrete optimization problems [104], which we refer to as the Quantum Adiabatic Algorithm[8] (QAA). Solving some discrete optimization problem is achieved by encoding the problem as the final Hamiltonian H_1 such that its groundstate is the optimal solution and choosing an initial Hamiltonian H_0 where the groundstate is known, easy to construct, and does not commute with H_1. Most common is the choice $H_0 = -\sum_{i=1}^{n} \sigma_i^x$ with groundstate $|+\rangle = \sqrt{2^{-n}} \sum_{i=0}^{2^n-1} |i\rangle$, which we can prepare easily.

The caveat with QAA is that we require the evolution time to be at most polynomial in the input size n but knowing the instantaneous eigenenergies of the time-dependent Hamiltonian $H(t)$ is often as challenging as solving the optimization problem itself. It is therefore unknown for many problems what kind of speedup is possible, or if there even is a speedup, over classical

[8]The quantum adiabatic algorithm is used synonymously with the term adiabatic quantum computation.

algorithms.

The issue of analyzing the instantaneous eigenenergies has resulted in the approximate version of QAA called Quantum Annealing (QA), where the evolution time is not guaranteed to ensure adiabatic evolution. One resulting difference between QA and QAA is that we now consider the overlap $p = |\langle\psi^*|\psi\rangle|^2$ between the state $|\psi\rangle$ we have obtained after evolving the system for some time T and the desired solution $|\psi^*\rangle$. By repeating the QA algorithm

$$k = \frac{\ln(1-p_{target})}{\ln(1-p)}$$

times, i.e., the process of initializing the system in the ground state of the initial Hamiltonian and evolving the system for some time T to the final Hamiltonian, the probability of finding the solution can be increased to p_{target}. To date, it is also unknown for many problems if QA can provide speedup or any significant advantage over classical algorithms, but it remains nonetheless an interesting heuristic to explore.

The Quantum Approximate Optimization Algorithm

QAOA is a variational hybrid quantum-classical algorithm, parameterized by the positive integer p that determines the depth of the quantum algorithm[9].

Although the algorithm is capable of universal quantum computing [105]–[108], the most common goal is to find approximate solutions to minimization (or maximization) problems. One property of QAOA discussed in [63] is that the solution quality is monotonically increasing with the parameter p, assuming the angles $\vec{\gamma}$ and $\vec{\beta}$ in Eq. (4.5) are optimal[10] and an ideal quantum computer. Furthermore, in [63] it was shown that under the assumption that the angles are small if $p \to \infty$ the algorithm becomes exact and finds the optimal solution (if the QAA can find the optimal solution).

A QAOA circuit creates, in an ideal setting, the state

$$|\vec{\gamma},\vec{\beta}\rangle = \prod_{i=0}^{p-1} U_m(\beta_{p-i}, H_m)U_c(\gamma_{p-i}, H_c)\,|\phi\rangle_{initial} \tag{4.5}$$

[9]The algorithm depth of QAOA is also called the number of layers in literature.

[10]The angles are required to be the optimal solution to the problem defined in Eq.(4.7)-(4.9). Hence for a minimization problem $M_p \leq M_{p-1}$.

where we, for each layer i, first apply the unitary cost operator U_c and second the mixer operator U_m that acts on n qubits. The most common choice of operators has been $U_m = e^{-i\beta H_m}$ and $U_c = e^{-i\gamma H_c}$, where we refer to H_m and H_c as the mixer and cost Hamiltonian, respectively. Since NP-complete problems can be encoded into an Ising spin glass Hamiltonian [109], the cost Hamiltonian is often given explicitly in this form, that is, $H_c = \sum_{i\in V} h_i\sigma_i^z + \sum_{\{i,j\}\in E} J_{ij}\sigma_i^z\sigma_j^z$. Moreover, we note that the goal of finding the solution to a minimization problem is the same as the goal of finding the ground state of the cost Hamiltonian. Here, we view H_c as representing an undirected graph $G = (V, E)$ with vertex set V and edge set E. For each edge $\{i, j\} \in E$, there exists a weight J_{ij}, and for each vertex $i \in V$ there is an associated weight h_i. In some ways, this makes various problems defined on undirected graphs particularly intuitive since we must, for example, translate an integer linear program to the Ising spin glass Hamiltonian, and it is not always straightforward how properties of the integer linear program are connected to graph properties. For example, we can state that we consider QAOA for a MaxCut problem with three regular graphs and the underlying graph to the cost Hamiltonian has that specified property.

Next, we will discuss the choice of mixer Hamiltonian. The choice is, in principle, free but the mixer operator should somehow be capable of connecting the initial state to states we accept as a solution. The most commonly[11] seen mixer Hamiltonian is $H_m = \sum_{i=1}^n \sigma_i^x$ and a natural initial state is then $|+\rangle = \sqrt{2^{-n}}\sum_{i=0}^{2^n-1}|i\rangle$. However, other choices of mixer Hamiltonian and initial state can be beneficial, as discussed in [110], by restricting QAOA into some subspace of the whole Hilbert space.

By considering the expectation value function

$$E_p(\vec{\gamma}, \vec{\beta}) = \langle\vec{\gamma}\vec{\beta}|H_c|\vec{\gamma}\vec{\beta}\rangle \tag{4.6}$$

[11]When choosing this mixer Hamiltonian, the algorithm is sometimes called vanilla QAOA and refers to the paper by Farhi et al. [63]. Furthermore, the initial state chosen with this mixer Hamiltonian is $|+\rangle$ and is the ground state of $-H_m$ that is typically the initial Hamiltonian in the Adiabatic Quantum Algorithm.

and the non-linear continuous optimization problem

$$\begin{aligned} M_p = \text{minimize} \quad & E_p(\vec{\gamma}, \vec{\beta}), && (4.7) \\ \text{subject to} \quad & \vec{\gamma} \in D_\gamma, && (4.8) \\ & \vec{\beta} \in D_\beta, && (4.9) \end{aligned}$$

for some fixed p, the probability of obtaining a string that is either the optimum or some distance away when measuring is high provided that p is sufficiently large and that the optimization problem in Eq. (4.7)-(4.9) can be solved. Thus, if a measurement is performed in the computational basis, we obtain a solution candidate string $\vec{z} \in \{-1, +1\}^n$, which can be evaluated for the cost Hamiltonian. I.e., if the process of (1) constructing the QAOA state with optimal angles and (2) measuring the state in the computational basis is repeated sufficiently many times, we should obtain a solution string that is near the expectation value function for the fixed angles.

We also note that for the mixer operator $U_m = \prod_{i=1}^{n} e^{-i\beta\sigma_i^x}$ the domain defined in Eq. (4.9) becomes $[0, \pi]^{\times p}$ and if the cost Hamiltonian has integer eigenvalues the domain in Eq. (4.8) is $[0, 2\pi]^{\times p}$. A priori, what depth of the

Algorithm 1 QAOA

```
Input: $p \geq 1$
  $(\vec{\gamma}, \vec{\beta}) \leftarrow$ solve Eq. (4.7)-(4.9)
  Construct the state $|\vec{\gamma}, \vec{\beta}\rangle$
  Measure $|\vec{\gamma}, \vec{\beta}\rangle$ in the computational basis
  Repeat two former steps $N$ times
Output: Best solution string found
```

algorithm is sufficient is not known for many optimization problems, which can cause issues when noise is present in the system. In practice, we might therefore need to increase the algorithm depth until some condition holds. However, the QAOA algorithm can be simply stated as in Alg. 1.

The issue of solving Eq. (4.7)-(4.9) appears daunting, as this problem is NP-hard [111] and we plan to solve it with a classical algorithm that queries a quantum computer to get values $E_p(\vec{\beta}, \vec{\gamma})$ for angles in their respective domains.

We can consider a few different solution approaches. One approach is to obtain a closed-form expression of the expectation value function. In that case, we can either find good angles by numerical optimization methods without a quantum computer or possibly determine good angles without using numerical optimization techniques by analyzing the closed-form expression [63], [112]. A second approach is to approximately simulate a quantum computer with, for example, matrix product states and tensor networks as in [113]. A third approach is to utilize machine learning techniques [114]–[116] and other numerical optimization techniques where each query of the expectation value function is obtained by using a quantum computer. Here there have been proposals that can reduce the number of queries we require by finding good angles for small instances and using them for larger instances and/or unseen instances and interpolating angles [117] for larger algorithm depths. Many of these proposals use the fact that the angles of QAOA appear to concentrate for certain problems and distributions, see [112], [118]. In this thesis, we have focused on the interpolation strategy, and note that since this problem is NP-hard, in general, we might not expect to have access to optimal angles.

At the time of writing this thesis, it seems that we are not sure if QAOA is capable of solving problems better and/or faster than the best classical algorithms. Although we have strong evidence that a classical computer can't simulate QAOA exactly [119], this does not say anything about what problems QAOA can solve or how much resources, with respect to time, the algorithm requires. Some interesting performance results have indeed shown that a classical algorithm outperforms QAOA or achieves the same approximation ratio for problems such as MaxCut [120]–[122] and MAX-3-XOR [123]. On the other hand, QAOA was observed to outperform a classical algorithm for MAX-k-XOR when $k > 4$ [124]. Furthermore, Farhi has been able to analyze QAOA extensively for Maximum Independent Set [125] and the Sherrington-Kirkpatrick model [112]. Results like these are vital for our understanding of the algorithm, and more such results are desirable. Notably, many of these results are restricted to constant algorithm depth or logarithmic depth in the number of qubits n. Negative results with restricted algorithm depth and restricted graphs are thus not excluding QAOA from outperforming classical algorithms for greater algorithm depths and other graph structures, and the ultimate question of whether QAOA can be advantageous is still open.

Finally, we would like to mention the issue of noise for QAOA [126]–[130]. If we indeed need to have logarithmic or polynomial algorithm depth, as we increase the algorithm depth, noise will be more important to consider as shown in [131], [132] where noise deprecates QAOA's performance. It also seems likely that establishing error mitigation techniques can be helpful or required [133].

CHAPTER 5

Summary of Papers

In this chapter, we give a summary of the appended papers. Papers A, C, and D are concerned with the algorithm QAOA. Paper B, in contrast, is related to the algorithm Quantum Annealing.

5.1 Paper A

In this work, we proposed a hybrid quantum-classical heuristic algorithm that augments the classical Branch-and-Price algorithm. Branch-and-Price is augmented in a similar fashion as in [134] with the classical integer program solver PAQS [135]. The main distinction here is that we propose to use a quantum algorithm to solve the current integer program, which is the integer version of the RMP. Although the heuristic is believed to be useful for several real-world problems as it is tied to Branch-and-Price, we naturally explored the method for extracted and simplified Tail Assignment RMP instances, as the focus of this thesis is quantum algorithms for airline scheduling problems.

Consequently, the problems we considered were both Exact Cover and Set Partitioning, and QAOA was the considered quantum algorithm. The results were obtained for an ideal quantum computer via numerical simulations of

QAOA circuits. It was found that balancing the objective and constraint parts of the Hamiltonian is important to reach a better performance for QAOA when attempting to solve Set Partitioning and that setting the penalty unnecessarily high can lead to an increased requirement on the algorithm depth.

It was also found for Exact Cover that QAOA, in general, requires lower algorithm depth as the number of feasible solutions increases. This coincides with the fact that the average node degree of the underlying graph decreases. In particular, we also observed this effect for the Set Partitioning problem, where we only accepted the optimal solution. In Paper C, it was found that a higher average node degree coincided with a worse performance of QAOA. This means that the numerical results in both papers point to the fact that the node degree can affect the performance of QAOA.

5.2 Paper B

This paper evaluated RMP instances extracted from Tail Assignment ranging from small to intermediate size for the Quantum Annealing algorithm on the D-wave machines Advantage and 2000Q. 2000Q and Advantage were compared for instances up to 100 decision variables, which is considerably larger than the instances we studied in Paper A. Instances with 120 decision variables were also studied with Advantage, but not possible to solve with 2000Q. The instances were both sparsely connected and close to fully connected, allowing us to analyze how the graph density and instance size affect the performance of both machines.

The results show that the new and larger machine Advantage solves the integer program instances in close to half the time required by 2000Q, with respect to programming and readout time. In Fig. 3, the annealing time is varied from 1-2000 μs against the success rate, for which Advantage outperforms 2000Q for most of the instances (with the exception of some of the smaller sparse graphs), meaning that the annealing time is shorter for Advantage compared to 2000Q. The results indicate that the connectivity of the machine's topology, which is higher in Advantage compared to 2000Q, is one important factor that enables Advantage to be superior to 2000Q. Thus, showing that quantum annealing could be useful in practice when the hardware is scaled up in size.

5.3 Paper C

Here, we studied the success probability of QAOA for Exact Cover instances with exactly one solution derived from Tail Assignment. The results were obtained for an ideal quantum computer via numerical simulations of quantum circuits. It was shown that the interpolation strategy presented in [117] could be utilized for the Exact Cover instances and that QAOA could, in the ideal case, give near unit success probability for an algorithm depth that was smaller than the number of qubits (i.e., instance sizes). It was also found that the performance of QAOA decreased for instances with a high average node degree compared to instances with a lower average node degree.

5.4 Paper D

Here, we implemented QAOA on a quantum processor consisting of superconducting transmon qubits for Exact Cover instances with two decision variables. We experimentally investigated the algorithm and processor for one and two layers, demonstrating that the success probability increased, as expected, as the algorithm depth increased. The maximum probability obtained was 96.6% for algorithm depth 2, where theory predicted 96.3% when gate fidelities were considered, compared to the ideal case, which predicted 100% success probability. Thus, the results show agreement between experiments and theory in the energy landscapes and algorithm performance, indicating low error rates.

CHAPTER 6

Concluding Remarks and Future Work

This chapter summarizes the conclusions, starting with Paper A and C since they are highly connected. We then give the conclusions for Paper B and D that consider existing devices. Finally, we discuss future opportunities related to Multi-Commodity Network Flow problems.

6.1 Paper A and C

Although the numerical results indicate that we, in many cases, can find feasible solutions and even the optimal solution for small instances of Set Partitioning and Exact Cover with polynomial algorithm depth, we recognize that these sizes are orders of magnitude smaller than the problems solved in practice. Our results can, therefore, not be compared to classical solvers in any meaningful way yet, nor can they arbitrarily be extrapolated to larger instance sizes.

To understand what algorithm depth is required for larger instances, a larger quantum device and/or constructing a mathematical proof of the required algorithm depth is needed. This feat has been achieved in [125] and [136] for Maximum Independent Set, for example. Such problems fall into the category of Ising models where all h_i terms are zero and all edge terms J_{ij} are one. As we, and others as far as we know, have yet to be successful in analyzing the behavior of QAOA by analytical means in a more general setting, it would be a highly valuable result to obtain. One possible avenue to achieve this is to find characteristics in Set Partitioning and Exact Cover that are related to characteristics of the underlying Ising spin glass Hamiltonian graph. Another possible method is to explore if Exact Cover and Set Partitioning have the overlap gap property, as this is exploited in the proof for Maximum Independent Set.

Whilst understanding the performance of QAOA for a general Ising spin glass Hamiltonian in the ideal setting remains an important open question, there is one important aspect that can depreciate the performance of QAOA, and that is noise. Furthering the understanding of noise as in [126]–[130], gives more insight into if QAOA truly is, or can be, noise resilient.

Other variants of QAOA as the Quantum Alternating Operator Ansatz [132], warm starting QAOA or RQAOA can also be interesting to investigate. We can, for example, view RQAOA [122] as an error mitigation technique as it reduces the instance size in each iteration and can possibly shorten the algorithm depth, beyond the fact that some evidence has been presented that RQAOA also can outperform QAOA for a certain problem of any size. Introducing further constraints as is done in the Quantum Alternating Operator Ansatze by fixing the Hamming weight of the solution string can also prove to be fruitful for some problems. We believe that constructing new initial states and mixing operators will continue to be an interesting research direction.

Finally, it would be interesting to understand the amount of entanglement that exists in QAOA circuits, this has to some extent been studied in [137], [138], but remains an important open question.

6.2 Paper B

The benchmark results obtained from both Advantage and 2000Q demonstrated that Quantum Annealing machines can solve intermediate-sized integer programs. The results also showed that QA machines perform better when they are scaled up in size and have improved connectivity. However, we are still lacking knowledge regarding the instantaneous energy gap for problems such as Exact Cover. This could be a future research possibility as well as conducting further empirical studies for average cases of Set Partitioning, Exact Cover, and Set Cover. In particular, if we in the future can embed problems with nearly 1000 decision variables, more interesting and realistic distributions are possible to study.

6.3 Paper D

The demonstration of toy problems on a superconducting quantum processor showed the quality of the device. It does not, however, say anything about the performance of any problem of interest. A future research possibility could be to use larger systems available, e.g., IBM's quantum processor, and possibly introduce some error correcting scheme as discussed in [132].

6.4 Quantum Algorithms and Integer Network Flows

Thus far, we have focused on the near-term gate-based algorithm QAOA and quantum annealing to some extent. However, the nature of many airline scheduling problems is such that there exist a vast number of constraints, and the number of variables is large, both in a Branch-and-Price augmented scheme and in an arc-flow formulation. It might accordingly be worthwhile to question the usefulness of variational algorithms for such large problems in the long-term development of quantum computers. It can therefore be interesting to consider fault-tolerant algorithms, such as Montanari's Branch-and-Bound algorithm [139], or other algorithms that are based on Dürr and Hoyer's search algorithm [140]. Such algorithm ideas have been presented by Ambainis for maximum flow in networks in [141], which of course is not applicable to multi-commodity network flows at this point.

To summarize, more effort is required to understand simple multi-commodity network flow problems in relation to quantum algorithms in various decompositions as these problems model airline scheduling problems.

References

[1] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization.* Springer US, 2009, ISBN: 9780387747590.

[2] G. Yu and B. Thengvall, "Airline optimization," in *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds. Springer US, 2009, pp. 26–30, ISBN: 978-0-387-74759-0.

[3] P. Belobaba, A. Odoni, C. Barnhart, and P. Belobaba, *The Global Airline Industry* (Aerospace Ser). John Wiley & Sons, Incorporated, 2015, ISBN: 9781118881149.

[4] C. Barnhart and A. Cohn, "Airline schedule planning: Accomplishments and opportunities," *Manufacturing & Service Operations Management*, vol. 6, no. 1, pp. 3–22, 2004, ISSN: 15234614.

[5] A. E. E. Eltoukhy, F. T. S. Chan, and S. H. Chung, "Airline schedule planning: A review and future directions," *Industrial Management & Data Systems*, vol. 117, no. 6, pp. 1201–1243, 2017, ISSN: 02635577.

[6] S. Yan and H.-F. Young, "A decision support framework for multi-fleet routing and multi-stop flight scheduling," *Transportation Research Part A: Policy and Practice*, vol. 30, no. 5, pp. 379–398, 1996, ISSN: 0965-8564.

[7] K. Wei, V. Vaze, and A. Jacquillat, "Airline timetable development and fleet assignment incorporating passenger choice," *Transportation Science*, vol. 54, no. 1, pp. 139–163, 2020, ISSN: 00411655.

[8] S. Yan and C.-H. Tseng, "A passenger demand model for airline flight scheduling and fleet routing," *Computers & Operations Research*, vol. 29, no. 11, pp. 1559–1581, 2002, ISSN: 0305-0548.

[9] L. H. Lee, C. U. Lee, and Y. P. Tan, "A multi-objective genetic algorithm for robust flight scheduling using simulation," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1948–1968, 2007, ISSN: 0377-2217.

[10] S. Yan, C.-H. Tang, and M.-C. Lee, "A flight scheduling model for taiwan airlines under market competitions," *Omega*, vol. 35, no. 1, pp. 61–74, 2007, ISSN: 0305-0483.

[11] S. Yan, C.-H. Tang, and T.-C. Fu, "An airline scheduling model and solution algorithms under stochastic demands," *European Journal of Operational Research*, vol. 190, no. 1, pp. 22–39, 2008, ISSN: 0377-2217.

[12] J. Hai and C. Barnhart, "Dynamic airline scheduling," *Transportation Science*, vol. 43, no. 3, pp. 336–354, 2009, ISSN: 00411655.

[13] M. Sohoni, L. Yu-Ching, and D. Klabjan, "Robust airline scheduling under block-time uncertainty," *Transportation Science*, vol. 45, no. 4, pp. 451–464, 2011, ISSN: 00411655.

[14] H. Jiang and C. Barnhart, "Robust airline schedule design in a dynamic scheduling environment," *Computers and Operations Research*, vol. 40, no. 3, pp. 831–840, 2013, ISSN: 0305-0548.

[15] B. Kepir, Ç. Koçyiğit, I. Koyuncu, M. B. Özer, B. Y. Kara, and M. A. Gürbüz, "Flight-scheduling optimization and automation for anadolujet," *Interfaces*, vol. 46, no. 4, pp. 315–325, 2016, ISSN: 00922102.

[16] A. Levin, "Scheduling and fleet routing models for transportation systems," *Transportation Science*, vol. 5, no. 3, pp. 232–255, 1971, ISSN: 00411655.

[17] C. Barnhart, T. Kniker, and M. Lohatepanont, "Itinerary-based airline fleet assignment," *Transportation Science*, vol. 36, no. 2, pp. 199–217, 2002, ISSN: 00411655.

[18] J. Rosenberger, E. Johnson, and G. Nemhauser, "A robust fleet-assignment model with hub isolation and short cycles," *Transportation Science*, vol. 38, no. 3, pp. 357–368, 2004, ISSN: 00411655.

[19] H. Sherali, E. Bish, and X. Zhu, “Polyhedral analysis and algorithms for a demand-driven refleeting model for aircraft assignment,” *Transportation Science*, vol. 39, no. 3, pp. 349–366, 2005, ISSN: 15265447.

[20] N. Bélanger, G. Desaulniers, F. Soumis, J. Desrosiers, and J. Lavigne, “Weekly airline fleet assignment with homogeneity,” *Transportation Research Part B*, vol. 40, no. 4, pp. 306–318, 2006, ISSN: 0191-2615.

[21] B. Smith and E. Johnson, “Robust airline fleet assignment: Imposing station purity using station decomposition,” *Transportation Science*, vol. 40, no. 4, pp. 497–516, 2006, ISSN: 15265447.

[22] T. Jacobs, B. Smith, and E. Johnson, “Incorporating network flow effects into the airline fleet assignment process,” *Transportation Science*, vol. 42, no. 4, pp. 514–529, 2008, ISSN: 15265447.

[23] J. Dumas, F. Aithnard, and F. Soumis, “Improving the objective function of the fleet assignment problem,” *Transportation Research Part B*, vol. 43, no. 4, pp. 466–475, 2009, ISSN: 0191-2615.

[24] V. L. Pilla, J. M. Rosenberger, V. Chen, N. Engsuwan, and S. Siddappa, “A multivariate adaptive regression splines cutting plane approach for solving a two-stage stochastic programming fleet assignment model,” *European Journal of Operational Research*, vol. 216, no. 1, pp. 162–171, 2012, ISSN: 0377-2217.

[25] D. T. Sanchez, B. Boyacı, and K. G. Zografos, “An optimisation framework for airline fleet maintenance scheduling with tail assignment considerations,” *Transportation Research Part B*, vol. 133, pp. 142–164, 2020, ISSN: 0191-2615.

[26] J. Abara, “Applying integer linear programming to the fleet assignment problem,” *Interfaces*, vol. 19, no. 4, pp. 20–28, 1989, ISSN: 00922102.

[27] M. E. Berge and C. A. Hopperstad, “Demand driven dispatch. a method for dynamic aircraft capacity assignment, models and algorithms,” *Operations Research*, vol. 41, no. 1, pp. 153–168, 1993, ISSN: 0030364X.

[28] C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, and G. Sigismondi, “The fleet assignment problem: Solving a large-scale integer program,” English, *Mathematical Programming*, vol. 70, no. 2, pp. 211–232, 1995.

[29] K. Talluri, “Swapping applications in a daily airline fleet assignment,” *Transportation Science*, vol. 30, no. 3, pp. 237–248, 1996, ISSN: 00411655.

[30] G. Desaulniers, J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis, “Daily aircraft routing and scheduling,” *Management Science*, vol. 43, no. 6, pp. 841–855, 1997, ISSN: 00251909.

[31] R. A. Rushmeier and S. A. Kontogiorgis, “Advances in the optimization of airline fleet assignment,” *Transportation Science*, vol. 31, no. 2, p. 159, 1997, ISSN: 00411655.

[32] A. Jarrah, J. Goodstein, and R. Narasimhan, “Efficient airline refleeting model for the incremental modification of planned fleet assignments,” *Transportation Science*, vol. 34, no. 4, pp. 349–363, 2000, ISSN: 00411655.

[33] B. Rexing, C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy, “Airline fleet assignment with time windows,” *Transportation Science*, vol. 34, no. 1, pp. 1–20, 2000, ISSN: 00411655.

[34] J. Arabeyre, J. Fearnley, F. Steiger, and W. Teather, “The airline crew scheduling problem: A survey.,” *Transportation Science*, vol. 3, no. 2, pp. 140–163, 1969, ISSN: 00411655.

[35] C. Barnhart, A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser, and P. H. Vance, “Airline crew scheduling,” in *Handbook of Transportation Science*, R. W. Hall, Ed. Boston, MA: Springer US, 2003, pp. 517–560, ISBN: 978-0-306-48058-4.

[36] X. Wen, X. Sun, Y. Sun, and X. Yue, “Airline crew scheduling: Models and algorithms,” *Transportation Research Part E*, vol. 149, 2021, ISSN: 1366-5545.

[37] M. Deveci and N. Ç. Demirel, “A survey of the literature on airline crew scheduling,” *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 54–69, 2018, ISSN: 0952-1976.

[38] M. Grönkvist, *The tail assignment problem* (Ph.D. dissertation). Chalmers tekniska högskola, 2005, ISBN: 9172916451.

[39] O. Khaled, M. Minoux, V. Mousseau, S. Michel, and X. Ceugniet, “A compact optimization model for the tail assignment problem,” *European Journal of Operational Research*, vol. 264, no. 2, pp. 548–557, 2018, ISSN: 0377-2217.

[40] Z. Liang, Y. Feng, X. Zhang, T. Wu, and W. A. Chaovalitwongse, "Robust weekly aircraft maintenance routing problem and the extension to the tail assignment problem," *Transportation Research Part B*, vol. 78, pp. 238–259, 2015, ISSN: 0191-2615.

[41] S. J. Maher, G. Desaulniers, and F. Soumis, "The daily tail assignment problem under operational uncertainty using look-ahead maintenance constraints," *European Journal of Operational Research*, vol. 264, no. 2, pp. 534–547, 2018, ISSN: 0377-2217.

[42] M. Fuentes, L. Cadarso, V. Vaze, and C. Barnhart, "The tail assignment problem: A case study at vueling airlines," *Transportation Research Procedia*, vol. 52, pp. 445–452, 2021, ISSN: 2352-1465.

[43] L. Lettovsky, "Airline operations recovery: An optimization approach," Ph.D. dissertation, School of Industrial Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA, 1997.

[44] N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine, "Airline disruption management—perspectives, experiences and outlook," *Journal of Air Transport Management*, vol. 13, no. 3, pp. 149–162, 2007, ISSN: 0969-6997.

[45] J. Sakurai and J. Napolitano, *Modern Quantum Mechanics.* Sep. 2020, ISBN: 9781108473224.

[46] M. R. Garey and D. S. Johnson, *Computers and intractability : a guide to the theory of NP-completeness* (A series of books in the mathematical sciences). Freeman, 1979, ISBN: 0716710455.

[47] S. Arora and B. Barak, *Computational complexity : a modern approach.* Cambridge University Press, 2009, ISBN: 9780521424264.

[48] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation. : Combinatorial Optimization Problems and Their Approximability Properties.* Springer Berlin Heidelberg, 1999, ISBN: 9783642584121.

[49] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff," *IEEE Solid-State Circuits Society Newsletter, Solid-State Circuits Society Newsletter, IEEE, IEEE Solid-State Circuits Soc. Newsl*, vol. 11, no. 3, pp. 33–35, 2006, ISSN: 1098-4232.

[50] G. Moore, “Progress in digital integrated electronics [technical literaiture, copyright 1975 ieee. reprinted, with permission. technical digest. international electron devices meeting, ieee, 1975, pp. 11-13.],” *IEEE Solid-State Circuits Society Newsletter, Solid-State Circuits Society Newsletter, IEEE, IEEE Solid-State Circuits Soc. Newsl*, vol. 11, no. 3, pp. 36–37, 2006, ISSN: 1098-4232.

[51] P. Benioff, “The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines,” *Journal of Statistical Physics*, vol. 22, no. 5, pp. 563–591, 1980.

[52] Y. Manin, *Computable and Non-Computable (in Russian).* Moscow: Sovetskoye Radio, 1980.

[53] R. P. Feynman, “Simulating physics with computers,” *International journal of theoretical physics*, vol. 21, no. 6/7, pp. 467–488, 1982.

[54] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information.* Cambridge University Press, 2010, ISBN: 9781107002173.

[55] J. D. Hidary, *Quantum Computing: An Applied Approach.* Springer International Publishing, 2019, ISBN: 9783030239220.

[56] D. Deutsch, “Quantum theory, the church–turing principle and the universal quantum computer,” *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, pp. 117–97, 1985.

[57] D. Deutsch and R. Jozsa, “Rapid solution of problems by quantum computation,” *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, pp. 553–558, 1992.

[58] E. Bernstein and U. Vazirani, “Quantum complexity theory,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[59] D. R. Simon, “On the power of quantum computation,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1474–1483, 1997.

[60] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[61] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, ISSN: 0001-0782.

[62] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219, ISBN: 0897917855.

[63] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, 2014. arXiv: `1411.4028`.

[64] A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, Jul. 2014.

[65] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[66] W. I-Lin, "Multicommodity network flows: A survey, part i: Applications and formulations," *International Journal of Operations Research*, vol. 15, no. 4, pp. 145–153, 2018, ISSN: 1813-713X.

[67] W. I-Lin, "Multicommodity network flows: A survey, part ii: Solution methods," *International Journal of Operations Research*, vol. 15, no. 4, pp. 155–173, 2018, ISSN: 1813-713X.

[68] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows : theory, algorithms and applications.* Prentice Hall, 1993, ISBN: 013617549X.

[69] R. Garfinkel and G. Nemhauser, *Integer programming* (Series in decision and control). John Wiley, n.d, ISBN: 0-471-29195-1.

[70] G. Dantzig, A. Orden, and P. Wolfe, "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, vol. 5, Jun. 1955.

[71] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960, ISSN: 0030364X.

[72] L. S. Lasdon, *Optimization theory for large systems.* Dover Publications, 2002, ISBN: 0486419991.

[73] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, Nov. 2005, ISSN: 0030-364X.

[74] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," *ECONOMETRICA*, vol. 28, no. 3, pp. 497–520, 1960.

[75] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, 2016, ISSN: 1572-5286.

[76] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, Feb. 1970.

[77] M. Jünger, M. Elf, and V. Kaibel, "Rotation planning for the continental service of a european airline," in *Mathematics — Key Technology for the Future: Joint Projects between Universities and Industry*, W. Jäger and H.-J. Krebs, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 675–689, ISBN: 978-3-642-55753-8.

[78] M. Daskin and N. Panayotopoulos, "A lagrangian relaxation approach to assigning aircraft to routes in hub and spoke networks," *Transportation Science*, vol. 23, pp. 91–99, May 1989.

[79] G. Desaulniers, J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis, "Daily aircraft routing and scheduling," *Management Science*, vol. 43, pp. 841–855, Jun. 1997.

[80] J.-F. Cordeau, G. Stojkovic, F. Soumis, and J. Desrosiers, "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation Science*, vol. 35, pp. 375–388, Nov. 2001.

[81] M. Bartholomew-Biggs, S. Parkhurst, and S. Wilson, "Global optimization approaches to an aircraft routing problem," *European Journal of Operational Research*, vol. 146, pp. 417–431, Apr. 2003.

[82] R. Gopalan and K. T. Talluri, "The aircraft maintenance routing problem," *Operations Research*, vol. 46, pp. 260–271, 1998.

[83] A. Sarac, R. Batta, and C. M. Rump, "A branch-and-price approach for operational aircraft maintenance routing," *European Journal of Operational Research*, vol. 175, pp. 1850–1869, 2006.

[84] Z. Liang, W. A. Chaovalitwongse, H. C. Huang, and E. L. Johnson, "On a new rotation tour network model for aircraft maintenance routing problem," *Transportation Science*, vol. 45, pp. 109–120, 2011.

[85] Z. Liang and W. A. Chaovalitwongse, "A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem," *Transportation Science*, vol. 47, pp. 493–507, 2013.

[86] N. M. Kabbani and B. W. Patty, "Aircraft routing at american airlines.," in *proceedings of the agifors symposium*, 1992.

[87] K. Talluri, "Swapping applications in a daily airline fleet assignment," *Transportation Science*, vol. 30, pp. 237–248, Aug. 1996.

[88] K. T. Talluri, "The four-day aircraft maintenance routing problem," *Transportation Science*, vol. 32, pp. 43–53, 1998.

[89] T. Feo and J. Bard, "Flight scheduling and maintenance base planning," *Management Science*, vol. 35, pp. 1415–1432, Dec. 1989.

[90] L. Clarke, E. Johnson, G. Nemhauser, and Z. Zhu, "The aircraft rotation problem," *Annals of Operations Research*, vol. 69, Jan. 1997.

[91] C. Sriram and A. Haghani, "An optimization model for aircraft maintenance scheduling and re-assignment," *Transportation Research Part A: Policy and Practice*, vol. 37, pp. 29–48, Jan. 2003.

[92] C. Barnhart, N. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, and R. G. Shenoi, "Flight string models for aircraft fleeting and routing," *Transportation Science*, vol. 32, pp. 208–220, 1998.

[93] J. Bard and I. Cunningham, "Improving through-flight schedules," *IIE Transactions*, vol. 19, pp. 242–251, Sep. 1987.

[94] A. Jarrah and J. Strehler, "An optimization model for assigning through flights," *IIE Transactions*, vol. 32, pp. 237–244, Mar. 2000.

[95] R. Ahuja, J. Goodstein, J. Orlin, and D. Sharma, "A very large-scale neighborhood search algorithm for the combined through-fleet-assignment model," *Massachusetts Institute of Technology (MIT), Sloan School of Management, Working papers*, vol. 19, Jan. 2003.

[96] R. K. Ahuja, J. Liu, J. Goodstein, A. Mukherjee, J. B. Orlin, and D. Sharma, “Solving multi-criteria through-fleet assignment models,” in *Operations Research in Space and Air*, T. A. Ciriani, G. Fasano, S. Gliozzi, and R. Tadei, Eds. Boston, MA: Springer US, 2003, pp. 233–256, ISBN: 978-1-4757-3752-3.

[97] R. Ahuja, J. Liu, J. Orlin, and J. Goodstein, “A neighborhood search algorithm for the combined through and fleet assignment model with time windows,” *Networks*, vol. 44, pp. 160–171, Sep. 2004.

[98] M. Danielsson and G. Karlsson, *The tail assignment problem for single and mixed aircraft fleets: Mathematical modelling, solution, and implementation* (master thesis). Chalmers tekniska högskola / Institutionen för matematiska vetenskaper, 2018.

[99] D. Deutsch, “Quantum computational networks,” *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 425, no. 1868, pp. 73–90, 1989, ISSN: 00804630.

[100] A. Chi-Chih Yao, “Quantum circuit complexity,” in *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, 1993, pp. 352–361.

[101] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, “Strengths and weaknesses of quantum computing,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1510–1523, Oct. 1997.

[102] T. Albash and D. A. Lidar, “Adiabatic quantum computation,” *Reviews of Modern Physics*, vol. 90, no. 1, Jan. 2018.

[103] A. Messiah, *Quantum mechanics.* North-Holland, 1961.

[104] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic evolution*, 2000. arXiv: `quant-ph/0001106`.

[105] S. Lloyd, *Quantum approximate optimization is computationally universal*, 2018. arXiv: `1812.11075`.

[106] M. E. S. Morales, J. D. Biamonte, and Z. Zimborás, “On the universality of the quantum approximate optimization algorithm,” *Quantum Information Processing*, vol. 19, no. 9, Aug. 2020.

[107] J. C. Aguma, *An upper bound on the universality of the quantum approximate optimization algorithm*, 2021. arXiv: `2104.01993`.

[108] H. Zheng, Z. Li, J. Liu, S. Strelchuk, and R. Kondor, *Speeding up learning quantum states through group equivariant convolutional quantum ansätze*, 2021. arXiv: `2112.07611`.

[109] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014.

[110] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, Feb. 2019.

[111] L. Bittel and M. Kliesch, "Training variational quantum algorithms is NP-hard," *Physical Review Letters*, vol. 127, no. 12, Sep. 2021.

[112] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, "The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size," *Quantum*, vol. 6, p. 759, Jul. 2022.

[113] M. Streif and M. Leib, *Training the quantum approximate optimization algorithm without access to a quantum processing unit*, 2019. arXiv: `1908.08862`.

[114] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2367–2375, Apr. 2020.

[115] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, *Reinforcement-learning-based variational quantum circuits optimization for combinatorial problems*, 2019. arXiv: `1911.04574`.

[116] D. Wecker, M. B. Hastings, and M. Troyer, "Training a quantum optimizer," *Physical Review A*, vol. 94, no. 2, Aug. 2016.

[117] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, Jun. 2020, ISSN: 2160-3308.

[118] F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, *For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances*, 2018. arXiv: `1812.04170`.

[119] E. Farhi and A. W. Harrow, *Quantum supremacy through the quantum approximate optimization algorithm*, 2016. arXiv: 1602.07674.

[120] M. B. Hastings, *Classical and quantum bounded depth approximation algorithms*, 2019. arXiv: 1905.07047.

[121] K. Marwaha, "Local classical max-cut algorithm outperforms p=2 qaoa on high-girth regular graphs," *Quantum*, vol. 5, p. 437, Apr. 2021.

[122] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, "Obstacles to variational quantum optimization from symmetry protection," *Physical Review Letters*, vol. 125, no. 26, Dec. 2020.

[123] B. Barak, A. Moitra, R. O'Donnell, *et al.*, *Beating the random assignment on constraint satisfaction problems of bounded degree*, 2015. arXiv: 1505.03424.

[124] K. Marwaha and S. Hadfield, "Bounds on approximating max kxor with quantum and classical local algorithms," *Quantum*, vol. 6, p. 757, Jul. 2022.

[125] E. Farhi, D. Gamarnik, and S. Gutmann, *The quantum approximate optimization algorithm needs to see the whole graph: A typical case*, 2020. arXiv: 2004.09002.

[126] J. R. McClean, M. E. Kimchi-Schwartz, J. Carter, and W. A. de Jong, "Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states," *Physical Review A*, vol. 95, p. 042 308, 4 Apr. 2017.

[127] C. Xue, Z.-Y. Chen, Y.-C. Wu, and G.-P. Guo, *Effects of quantum noise on quantum approximate optimization algorithm*, 2019. arXiv: 1909.02196.

[128] J. Marshall, F. Wudarski, S. Hadfield, and T. Hogg, "Characterizing local noise in QAOA circuits," *IOP SciNotes*, vol. 1, no. 2, p. 025 208, Aug. 2020.

[129] E. Fontana, N. Fitzpatrick, D. M. Ramo, R. Duncan, and I. Rungger, "Evaluating the noise resilience of variational quantum algorithms," *Physical Review A*, vol. 104, p. 022 403, 2 Aug. 2021.

[130] J. Kattemölle and G. Burkard, *Effects of correlated errors on the quantum approximate optimization algorithm*, 2022. arXiv: 2207.10622.

[131] G. Quiroz, P. Titum, P. Lotshaw, *et al.*, *Quantifying the impact of precision errors on quantum approximate optimization algorithms*, 2021. arXiv: `2109.04482`.

[132] M. Streif, M. Leib, F. Wudarski, E. Rieffel, and Z. Wang, "Quantum algorithms with local particle-number conservation: Noise effects and error correction," *Physical Review A*, vol. 103, no. 4, Apr. 2021.

[133] R. Shaydulin and A. Galda, "Error mitigation for deep quantum optimization circuits by leveraging problem symmetries," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, Oct. 2021.

[134] T. Gustafsson, *A heuristic approach to column generation for airline crew scheduling* (Lic. dissertation). Chalmers tekniska högskola, 1999.

[135] D. Wedelin, "An algorithm for large scale 0–1 integer programming with application to airline crew scheduling," *Annals of Operations Research*, vol. 57, no. 1, pp. 283–301, 1995.

[136] E. Farhi, D. Gamarnik, and S. Gutmann, *The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples*, 2020. arXiv: `2005.08747`.

[137] R. Sreedhar, P. Vikstål, M. Svensson, A. Ask, G. Johansson, and L. García-Álvarez, *The quantum approximate optimization algorithm performance with low entanglement and high circuit depth*, 2022. arXiv: `2207.03404`.

[138] M. Dupont, N. Didier, M. J. Hodson, J. E. Moore, and M. J. Reagor, "Calibrating the classical hardness of the quantum approximate optimization algorithm," *PRX Quantum*, vol. 3, no. 4, Dec. 2022.

[139] A. Montanaro, "Quantum speedup of branch-and-bound algorithms," *Physical Review Research*, vol. 2, no. 1, Jan. 2020.

[140] C. Durr and P. Hoyer, *A quantum algorithm for finding the minimum*, 1996. arXiv: `quant-ph/9607014`.

[141] A. Ambainis and R. Spalek, *Quantum algorithms for matching and network flows*, 2005. arXiv: `quant-ph/0508205`.