# Design and application of quantum algorithms for railway optimisation problems

Camille Grange



# THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

**En Informatique**

**École doctorale I2S Information, Structures, Systèmes**

**Unité de recherche LIRMM, UMR 5506**

# Design and application of quantum algorithms for railway optimisation problems


**Présentée par Camille GRANGE**


**Le 09/07/2024**

**Sous la direction de Michaël POSS**

**Devant le jury composé de**

| | |
|---|---|
| Christoph DURR, Directeur de Recherche, CNRS, LIP6 | Rapporteur |
| Francesca GUERRIERO, Full Professor, Université de Calabria | Rapportrice |
| Sourour ELLOUMI, Professeure, ENSTA Paris | Examinatrice (Présidente du jury) |
| Giacomo NANNICINI, Associate Professor, Université de Californie du Sud | Examinateur |
| Michaël POSS, Directeur de Recherche, CNRS, LIRMM | Directeur de thèse |
| Eric BOURREAU, Maître de Conférence, Université de Montpellier | Co-encadrant |
| Valentina POZZOLI, Docteure, SNCF SA | Co-encadrante |

UNIVERSITÉ DE MONTPELLIER

**Abstract**

This thesis is dedicated to the conception and application of quantum algorithms for railway combinatorial optimization problems. Today, the optimization problems that SNCF faces are complex, often prohibiting finding the optimal solution for industrial instances with classical methods within a reasonable amount of time. Quantum computing is expected to improve the quality of solutions and reduce the computation time for some of these problems. Quantum algorithms for optimization are divided into two classes: exact algorithms and heuristics. The former demonstrate theoretical advantages for several problems but cannot be implemented on current quantum machines because they require too high-quality quantum resources. On the contrary, the latter can be implemented, at least for small instances, but there are no performance guarantees or proven quantum advantages yet. In this thesis, we analyze and propose algorithms that belong to each of these two classes.

On the one hand, we study the Variational Quantum Algorithms, which belong to the class of heuristics. These are hybrid quantum-classical algorithms that alternate between a parametrized quantum circuit and a classical optimizer. They allow solving unconstrained problems with binary variables, and we propose a general method to reformulate constrained integer problems into such problems. We highlight some properties of Variational Quantum Algorithms necessary for potential theoretical guarantees. In particular, we study QAOA (Quantum Approximate Optimization Algorithm) in light of the previous properties, and we provide a universal decomposition of the quantum circuit for problems whose objective function is polynomial. We solve with this algorithm a railway timetabling problem of SNCF. It consists of finding the transportation plan maximizing the operating profit according to the customers' demand taking into account the availability and cost of both the network and the rolling stock. To solve it with QAOA, we propose two simplifications with different adaptations of the original problem.

On the other hand, we design exact quantum-classical algorithms for two broad families of combinatorial problems. The first family relates to scheduling problems. We propose an algorithm that tackles a large class of NP-hard single-machine scheduling problems, which satisfy a specific dynamic programming property (Dynamic Programming Across the Subsets). Our algorithm, based on the seminal idea of Ambainis et al. (2019), combines classical dynamic programming and quantum search of the minimum in a table (generalization of Grover Search). It reduces the worst-case time complexity, sometimes at the cost of an additional pseudo-polynomial factor. We extend this algorithm to the 3-machine flowshop problem, also leading to a reduction of the complexity. The second family concerns robust optimization problems where the uncertainty set is a polytope. We present an algorithm that, relying on the classical method that deals with these problems, replaces some computations with quantum subroutines to achieve a speedup. Specifically, we study the two following quantum subroutines: the search of the minimum in a table and the resolution of a linear system.

**Résumé**

Cette thèse est dédiée à la conception et à l'application d'algorithmes quantiques pour la résolution de problèmes d'optimisation combinatoire ferroviaires. Aujourd'hui, les problèmes d'optimisation auxquels fait face la SNCF sont complexes, empêchant souvent une résolution à l'optimalité via des méthodes classiques en un temps raisonnable. L'informatique quantique est pressentie pour améliorer la qualité des solutions et diminuer le temps de calcul pour certains de ces problèmes. Actuellement, les algorithmes quantiques pour l'optimisation se divisent en deux classes : les algorithmes exacts et les heuristiques. Les premiers présentent un avantage théorique pour plusieurs problèmes, mais ne sont pas implémentables sur les machines actuelles car trop gourmands en ressources. Les seconds sont implémentables dès aujourd'hui, au moins sur de petites instances, ouvrant la porte aux premières applications, bien qu'ils ne présentent pas encore de garanties de performances ni d'avantage quantique. Dans cette thèse, nous analysons et proposons des algorithmes qui appartiennent à chacune de ces deux classes.

D'une part, nous étudions une classe d'heuristiques appelée Algorithmes Variationnels Quantiques. Il s'agit d'algorithmes hybrides quantique-classique, qui alternent entre l'exécution d'un circuit quantique paramétré et l'optimisation classique des paramètres. Ils permettent de résoudre des problèmes non contraints à variables binaires, et nous proposons une méthode générale pour reformuler des problèmes contraints à variables entières sous cette forme. Nous présentons certaines propriétés des Algorithmes Variationnels Quantiques, nécessaires pour envisager des preuves théoriques de garanties de performances. En particulier, nous étudions QAOA (Quantum Approximate Optimization Algorithm) en l'analysant à la lumière des précédentes propriétés et en donnant une décomposition universelle de son circuit quantique pour des problèmes dont la fonction objectif est polynomiale. Avec cet algorithme, nous résolvons un problème de conception de plan de transport de la SNCF. Ce problème a pour but de trouver un plan de transport qui correspond au meilleur compromis économique entre les bénéfices générés par la vente de billets aux voyageurs et les coûts d'exploitation, tout en respectant la disponibilité du réseau ferroviaire. Pour résoudre ce problème avec QAOA, nous proposons deux simplifications correspondant à différentes adaptations du problème métier initial.

D'autre part, nous élaborons des algorithmes quantiques-classiques exacts pour deux grandes familles de problèmes combinatoires. La première famille concerne les problèmes d'ordonnancement. L'algorithme proposé s'applique à une large classe de problèmes d'ordonnancement à une machine, NP-difficiles, qui satisfont une propriété de programmation dynamique particulière (Dynamic Programming Across the Subsets). L'algorithme, reprenant l'idée de Ambainis et al. (2019), allie la programmation dynamique classique et l'algorithme quantique de recherche du minimum dans une table (basé sur l'algorithme de Grover). Il permet de réduire la complexité en temps pire-cas, parfois au détriment de l'introduction d'un terme pseudo-polynomial. Nous étendons cet algorithme au problème du flowshop à trois machines, pour lequel une accélération est aussi obtenue. La deuxième famille relève des problèmes d'optimisation robuste où l'ensemble d'incertitude est un polytope. Nous proposons un algorithme qui, partant de l'algorithme classique traitant de ces problèmes, remplace certaines opérations par des routines quantiques afin d'obtenir une accélération. Précisément, nous étudions l'utilisation des deux routines quantiques suivantes : la recherche du minimum dans une table et la résolution d'un système linéaire.

**Remerciements**

Cette thèse est le fruit d'un travail collaboratif, sur le plan privé comme professionnel, qui m'a entièrement comblée pendant ces trois années.

Un immense merci et une grande reconnaissance envers mon directeur de thèse Michaël Poss, qui m'a formée en temps que chercheuse, au niveau scientifique comme aux niveaux éthique et humain. Un très grand merci à Eric Bourreau et Valentina Pozzoli, pour leur encadrement de qualité, leur soutien sans faille et leur bonne humeur !

Au delà de mon encadrement, j'ai eu la chance de travailler avec de nombreuses personnes bienveillantes et passionnées, sur mes travaux de thèse (Vincent T'kindt, Olivier Ploton, David de Almeida, François Ramond et bien d'autres) comme sur les enseignements donnés (Dimitri Watel, Antoine Genitrini, Mehdi Naima).

Je suis ravie d'avoir partagé ces trois années avec mes collègues et amis de la SNCF (Hugo, Louis, Jean, Charlie, Ariane, Lina, Héloïse...) ainsi que ceux du LIRMM (Imran, Mariam, Igor...). Nos discussions sur le cinéma, la cuisine, le jardinage, la littérature et autres arts ont été un vrai plaisir (et une nécessité !).

Enfin, je remercie mes proches de m'avoir soutenue pendant toute cette période. Guigui, pour sa présence fantastique au quotidien; mon frère, pour son aide et ses bonnes bouteilles; mes parents, pour m'avoir encouragée pendant ces trois années, et ce, avec autant d'intensité que pendant les 23 précédentes.

# Contents

# 1

# Introduction

This thesis is dedicated to the use of quantum algorithms to tackle combinatorial optimization problems, specifically addressing challenging railway problems whose resolutions remain difficult using current classical methods. The difficulty in solving these problems stems both from the inherent combinatorial complexity, often NP-hard, and from the industrial scale of the instances involved. Facing these obstacles, quantum computing is expected to both improve solution quality and reduce computation time in finding good solutions. Before the idea of considering quantum computing to solve optimization problems, these two disciplines initially followed separate paths before converging later on in the 1990s.

On the one hand, quantum computing traces its origins back to the early 1980s when the physicist Richard Feynman proposed the idea of simulating quantum systems using computers based on quantum mechanics. From this was born the notion of quantum algorithms: manipulating information using quantum bits (qubits), leveraging the principles of quantum mechanics. The first demonstration of its huge potential occurred in 1994 when Peter Shor developed a quantum algorithm to factor large numbers exponentially faster than classical computers (Shor, 1994).

On the other hand, combinatorial optimization dates back to more ancient times but its study intensified in the mid-20th century with the rise of operations research and computer science. The field witnessed significant advancements in numerical applications in the last seven decades with the development of metaheuristics, and efficient branch-and-cut algorithms to solve problems under the form of mixed-integer linear programs.

These two domains met for the first time with Grover Search (Grover, 1996), enabling a quadratic speedup for the search of an element in an unstructured database and opening the doors to the conception of new algorithms. Grover Search became a fundamental quantum subroutine, among others such as the Quantum Phase Estimation or the Quantum Fourier Transform, lying at the core of a large variety of quantum algorithms. In recent years, technical advances in quantum computers' construction have increasingly drawn the operational research community towards this new computing paradigm, trying to obtain numerical results, which brings us to the motivation of this thesis.

SNCF is the main French railway company. It transports in total more than 5 million passengers on its 35 000 km of railway network every day. Specifically focusing on high-speed trains, approximately 122 million passengers use them each year. Consequently, SNCF faces optimization problems that are up to its significant role in French society: mapping and servicing the territory on the one hand, and enabling a sustainable and necessary commitment to the ongoing ecological crisis on the other hand. A significant challenge is thus to deliver optimal service to customers while remaining competitive in the liberalized economy of train operators. The SNCF optimization problems are divided into three main categories. The first one contains *strategical* problems that structure all the other problems, often planned several years in advance. The second one concerns *pre-operational* problems, meaning adaptations to operations, or assignation of resources (agents, carriages) to the resulting planning of strategical problems' resolutions. They are done in a time frame ranging from months to days in advance. The third one concerns *real-*

*time* problems, meaning adaptations or changes to operations when a disruption or an unforeseen event occurs.

The difficulty of solving these combinatorial problems today comes from two major reasons. First, the limited time allowed for the resolution, mainly concerning real-time problems for which a decision must be taken quickly. Second, the consideration of large models. Today, classical resolutions of some problems have reached their limits, making it hard to find the optimal solution, or even a satisfying one, in a reasonable amount of time. Large problems are sub-divided into smaller problems, for example in the temporal or geographical perimeter, or creating intermediate steps of resolution into the original problem. Integrated problems are today solved into multiple successive problems, where the output solution of one corresponds to the input of the other. These necessary simplifications and *cuts* impact the quality of the solution. This motivates the search for new methods, among them quantum computing, to see *if*, *when*, and *how* could quantum algorithms improve these resolutions.

Currently, it is widely accepted that quantum computing should not disrupt the hierarchy of classical complexity classes. Specifically, quantum algorithms are not believed to solve NP-hard problems in polynomial time. They are rather envisaged to solve typical instances of some problems better than classical methods. Positive results comforting this latter idea already exist. Next, we provide an overview of these results, regarding both discrete and continuous optimization.

## 1.1 Optimization and quantum computing

There are two types of quantum algorithms for solving optimization problems. The first type concerns *exact* algorithms (i.e. output the optimal solution with a high probability of success). They provide theoretical speedups for several types of problems and algorithms, but are impossible to implement today because of the huge size of quantum resources they require. The second type encompasses heuristics, often designed today as hybrid quantum-classical algorithms, that can be implemented on current noisy quantum computers because the quantum part can be made rather small. The question of a quantum advantage using these methods is still open today.

### 1.1.1 Exact algorithms

As for classical algorithms, quantum algorithms are called *exact* when the sequence of steps at hand is proven to provide the optimal solution (in opposition to heuristics). However, the assertion that "the quantum algorithm finds the optimal solution" is always implicitly followed by "with high probability" due to the probabilistic behavior of quantum unitary operations. This means that the probability of success is strictly larger than $\frac{1}{2}$ and can get close to 1 by repeating the algorithm several times. Similarly, they are referred to as *bounded-error* algorithms, as for classical probabilistic algorithms.

Most exact algorithms are based on a restricted bunch of quantum subroutines. The most famous one is Grover Search (Grover, 1996), leading to various generalizations such as Amplitude Amplification (Brassard et al., 2002) or Quantum Minimum Finding (Dürr and Høyer, 1996), enabling quadratic speedup over several classical search algorithms. For instance, this latter generalization is used for dynamic programming, accelerating the resolution of the Traveling Salesman Problem and the Minimum Set Cover problem (Ambainis et al., 2019), the Steiner Tree problem (Miyamoto et al., 2020) or the Graph Coloring problem (Shimizu and Mori, 2022).

Another subroutine is the quantum walk lying at the core of algorithms that provide exponential speedups for some specific black-box problems (Childs et al., 2003) or polynomial speedups for

problems such as triangle problem (Magniez et al., 2007) or the element distinctness problem (Ambainis, 2007). Additionally, quantum walks are used for Backtracking algorithms (Montanaro, 2015), itself representing the subroutine for a quantum Branch-and-Bound achieving an almost quadratic speedup (Montanaro, 2020).

Two other emblematic subroutines are the Quantum Phase Estimation (Kitaev, 1995) and the Quantum Fourier Transform (Coppersmith, 2002). They are used, among others, for computing gradients (Gilyén et al., 2019) or solving linear systems (Harrow et al., 2009) with exponential speedups. The resolution of a linear system represents a quantum subroutine of numerous algorithms that *quantize* classical methods and achieve a polynomial speedup over them, such as the Simplex Method (Nannicini, 2021) or the Interior Point Method (Kerenidis and Prakash, 2020). The latter provides quantum algorithms for Linear Programming, Semi-Definite Programming, and Second-Order Cone Programming (Kerenidis et al., 2021).

### 1.1.2 Heuristics

The exact algorithms introduced above prove quantum advantages on numerous problems, but their implementations require a lot of quantum resources with high quality. Specifically, they need quantum computers with many qubits that can interact two by two, and quantum operations that can be applied in a row without generating noise. However, the current ones do not respect these criteria, encouraging the researchers to look into *lighter* algorithms, waiting for more powerful quantum computers to be built. They consist of hybrid algorithms that require both quantum and classical resources. Indeed, the classical part overcomes the limited and noisy quantum resources, whereas the quantum part still takes partial advantage of quantum information theory.

Such algorithms are metaheuristics, the most famous one being the Quantum Approximate Optimization Algorithm (QAOA) introduced by Farhi et al. (2014) to solve the MAX-CUT problem. QAOA takes place in a larger class of Variational Quantum Algorithms (Cerezo et al., 2021) which consist of alternating between a quantum circuit and a classical optimizer.

Another heuristic commonly used for solving combinatorial problems is the Quantum Annealing (Kadowaki and Nishimori, 1998). It maps the optimization problem to an energy operator and then uses the adiabatic theorem to approximate the operator's ground state, which is the optimal solution of the optimization problem.

QAOA and Quantum Annealing require the combinatorial optimization problem to be formulated as an Unconstrained Binary Optimization one, where the function is at most quadratic for Quantum Annealing but can be of higher polynomial degree for QAOA. Both are heuristics and have no performance guarantees for generic problems. Recently, several problems have been solved using these heuristics. Among them, *theoretical* problems such as MAX-CUT (Farhi et al., 2014), Travelling Salesman Problem (Ruan et al., 2020), MAX-3-SAT (Nannicini, 2019), Graph Coloring (Tabi et al., 2020) and Job Shop Scheduling (Kurowski et al., 2023). They are reformulated as QUBO and solved with QAOA on small instances. More *industrial* problems have also been tackled, such as knapsack problem for battery revenue (de la Grand'rive and Hullo, 2019) or smart charging of electric vehicles (Dalyac et al., 2021) with QAOA, or rolling stock planning (Bickert et al., 2021), aircraft loading (Pilon et al., 2021) or train seating arrangement (Gioda, 2021) with Quantum Annealing. However, due to the small size of instances processed today (imposed by the weak maturity of quantum computers) and to the nature of heuristics whose performances are evaluated empirically, no quantum advantage is emerging yet.

### 1.1.3 Quantum hardware

Since the idea proposed by Feynman in the 1980s of quantum computers, numerous quantum algorithms have been developed without any hardware available. Since the 2010s, quantum technologies have begun to emerge, making conceivable the construction of quantum computers powerful enough to apply the aforementioned algorithms. Let us provide a brief overview of the current state of quantum devices.

A quantum computer is defined by the physical implementation of the information unit manipulated during operations: the qubit. The nature of quantum operations follows in consequence. The main types of technologies for physical qubits currently being researched and developed are the following: superconductors, trapped ions, cold atoms, and photons. Each type has its unique advantages and challenges, and the question of which one will be the best in the future cannot be answered yet as the field continues to evolve rapidly.

To assess the quality of a given quantum computer, several aspects come into play. The first one concerns the qubits. Not only having a large number of qubits is important, but their quality also matters: the longer the coherence time (the time during which a qubit can maintain its quantum state without decoherence), the better the reliability of the quantum system. A second aspect is about the unit operations applied to the qubits: the error rate of the operation must be small, as well as the connectivity must be large (meaning that qubits can interact with many neighboring qubits). A third and major aspect is the scalability of a quantum system, referring to the ability to increase its size and complexity while maintaining its performance (quality of qubits and operations). Scalability is essential for realizing practical quantum computers capable of solving real-world problems.

Today, we are in an era called NISQ (Noisy Intermediate Scale Quantum) era (Preskill, 2018), characterized by the existence of quantum devices with a limited number of qubits and high error rates. These devices are not yet capable of achieving fault-tolerant quantum computation (FTQC), but they can still perform some quantum algorithms, such as heuristics for combinatorial optimization. Hopefully, they represent the beginning of the journey towards the FTQC era, where algorithms requiring significant resources, such as the exact algorithms mentioned above, could begin to be implemented. For this purpose, error mitigation and error correction are techniques being developed to improve the reliability and accuracy of quantum computations. The existence of a Quantum Random Access Memory (QRAM), enabling access to classical data by quantum queries, will be also necessary for many applications. However, practical implementations of QRAM remain a subject of ongoing research.

## 1.2 Outline of the thesis

This thesis is divided into three main parts, contributing to both heuristics and exact algorithms domains, with an introductory chapter covering the necessary concepts of quantum computing. The first part (Chapters 3 and 4) deals with variational quantum heuristics and their numerical applications to a railway timetabling problem of SNCF. The second part (Chapters 5 and 6) relates to the resolution of a broad class of scheduling problems with an exact quantum-classical algorithm combining Grover Search and dynamic programming. The third part (Chapter 7) also involves exact algorithms, but in order to address robust optimization problems with polyhedral uncertainty, using a Quantum Linear System algorithm as a subroutine. Next, we detail the content of each chapter.

Chapter 2 introduces the concepts of quantum computing and emphasizes the difference be-

tween classical and quantum computations. Specifically, we present all these notions in the gate-based model, also called the circuit model. Indeed, we will describe all the algorithms in this computation model throughout this manuscript.

In Chapter 3, we provide a mathematical description of Variational Quantum Algorithms (VQAs) and focus on one of them, the Quantum Approximate Optimization Algorithm (QAOA). VQAs are quantum-classical heuristics that alternate between a quantum circuit and a classical optimizer. They tackle unconstrained optimization problems of the form $\min_{x\in\{0,1\}^n} f(x)$, where $f$ is any function defined on $\{0,1\}^n$. More precisely, we describe the general class of VQAs, starting by defining the different parts that constitute and define a VQA, namely the quantum circuit, the classical optimizer, and the guiding function driving the classical optimization. Then, we characterize each part with properties that should be valid for potential theoretical guarantees. For the special case of QAOA, we describe the necessary reformulation of the initial problem into a Hermitian matrix and analyze it in light of the previous properties. We also provide a universal decomposition of the QAOA quantum circuit for the general case where $f$ is polynomial while only the quadratic case has been treated in the literature so far. We give a condensed overview of current empirical trends and theoretical limitations of QAOA. Finally, we present a generic method to tackle constrained integer optimization problems with VQAs by integrating the constraints as penalty terms in the objective function.

VQAs have the convenient property of an adjustable quantum circuits' depth, making them implementable on the current NISQ computers. Chapter 4, in which we address the railway timetabling problem for high-speed trains and solve it with QAOA, follows naturally. This challenging NP-hard problem for SNCF consists of finding the transportation plan maximizing the operating profit according to the customers' demand on the one hand, and the availability and cost of the network and the rolling stock on the other hand. First, we simplify it into two problems, a Set Cover Problem and an Extended Bin Packing problem, where the latter represents a closer version of the real-world problem. We reformulate each of them as an unconstrained binary problem, with either a polynomial or a quadratic objective function. We solve them with QAOA on small instances (simulating the quantum part on a classical simulator to avoid noise) to exhibit trends and limitations of numerical resolutions today. Furthermore, we compare numerically the reformulation that imposes the objective function to be quadratic with the one that does not.

In Chapter 5, we provide an exact quantum-classical algorithm to address a large class of scheduling problems that reduces their worst-case time complexity compared to the best-known classical methods, sometimes at the cost of an additional pseudo-polynomial factor. Our algorithm is adapting the seminal idea of Ambainis et al. (2019) that combines Quantum Minimum Finding (Dürr and Høyer, 1996) with dynamic programming to address NP-hard vertex ordering problems. We tackle NP-hard single-machine scheduling problems, for which we propose an extended version of Dynamic Programming Across the Subsets (DPAS) recurrences, widely used in the moderate exponential-time algorithms' literature. We consider two types of single-machine problems: one for which the dynamic programming is based on the *addition* of optimal values of the problem on sub-instances, and the other for which the dynamic programming allows the *composition* of sub-instances' values. We also address the 3-machine flowshop problem, for which the dynamic programming recurrence applies to a decision problem, resulting in a slightly different hybrid algorithm. Additionally, we provide an approximation scheme for the 3-machine flowshop problem, based on the hybrid algorithm, that disposes of the pseudo-polynomial factor in the time complexity. Throughout this chapter, we describe our hybrid algorithm, and its slight modifications, as it is usually done in the algorithmic quantum literature, namely with a high-

level description where quantum *boxes* interact with the classical part. We provide a rigorous and detailed description of the circuit-based implementation in Chapter 6.

Chapter 7 studies the use of quantum subroutines to solve robust MIN-MAX optimization problems where the uncertainty set is a polytope. The classical algorithm solving these problems (Omer et al., 2024) amounts to: a) formulating a set of problems according to the solutions to various linear systems and b) solving each of them at optimal and keeping the best solution. We propose to integrate two quantum subroutines into this classical algorithm. The first one is a straightforward application of Quantum Minimum Finding (Dürr and Høyer, 1996) for Step b), leading to a polynomial speedup of the worst-case time complexity. The second one is a quantum resolution of linear systems of Childs et al. (2017) for Step a). Due to the normalization of quantum states, among others, the quantum resolution introduces an error in the problems' formulations and consequently in the overall output of the classical algorithm. It seems that the error may not be controllable if one aims to achieve a speedup. However, the proposed way to use this quantum subroutine remains not the only one possible and thus is ongoing work.

This thesis concludes with a last chapter, Chapter 8, which summarizes the completed work and provides perspectives for future research.

# 2

# Basics tools for quantum computing

This chapter aims to provide all the necessary mathematical concepts and basic definitions of quantum computing to address combinatorial optimization problems. Specifically, we begin with a major definition from linear algebra, the tensor product, and with the *braket* notations. Next, we describe the various concepts of quantum computing in the gate-based model, underlying the differences with classical computations. We conclude by defining the complexity used to measure the performance of quantum algorithms.

## 2.1 Tensor product and braket notations

This section defines the tensor product, inner product and the braket notations. Notice that throughout the manuscript, we use the notations $[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$, and $[\![n_1, n_2]\!] := \{n_1, n_1 + 1, \dots, n_2\}$ for $n_1, n_2 \in \mathbb{N}$.

**Definition 2.1.1** (Tensor product)**.** *Let $n, m, p, q$ be integers, and let $A = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} \in \mathcal{M}_{n,m}(\mathbb{C})$ and $B = \begin{pmatrix} b_{11} & \dots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{p1} & \dots & b_{pq} \end{pmatrix} \in \mathcal{M}_{p,q}(\mathbb{C})$ be two complex matrices. Then, we define their tensor product, a bilinear operation, as*

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{pmatrix} \in \mathcal{M}_{np,mq}(\mathbb{C}),$$

*where* $a_{ij}B = \begin{pmatrix} a_{ij}b_{11} & \dots & a_{ij}b_{1q} \\ \vdots & \ddots & \vdots \\ a_{ij}b_{p1} & \dots & a_{ij}b_{pq} \end{pmatrix} \in \mathcal{M}_{p,q}(\mathbb{C})$, *for* $i \in [n], j \in [m]$.

**Proposition 2.1.2** (Tensor product properties)**.** *We draw attention to some useful properties of the tensor product:*

- $A \otimes B \neq B \otimes A$
- $(A \otimes B) \otimes C = A \otimes (B \otimes C) = A \otimes B \otimes C$
- $A \otimes (cB) = (cA) \otimes B = c(A \otimes B)$, *for* $c \in \mathbb{C}$

Let us next introduce the *braket* notation used in quantum computing.

**Definition 2.1.3** (Braket notation)**.** *Let* $\psi = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{2^n} \end{pmatrix} \in \mathbb{C}^{2^n}$ *be a column vector. We note* $|\psi\rangle$*, said "ket* $\psi$*", the vector* $\psi$ *itself. Thus, we define* $\langle\psi|$*, said "bra* $\psi$*", the conjugate transpose of* $|\psi\rangle$*. Specifically,*

$$|\psi\rangle = \psi = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{2^n} \end{pmatrix} ,$$
$$\langle\psi| = \bar{\psi}^T = \begin{pmatrix} \bar{\psi_1} & \dots & \bar{\psi_{2^n}} \end{pmatrix} .$$

*Thus, ket vectors are always column vectors whereas bra vectors are row vectors. With this notation, the norm of the vector* $\psi$ *is*

$$\langle\psi|\psi\rangle = \sum_{i=1}^{2^n} |\psi_i|^2 .$$

**Definition 2.1.4** (Inner product)**.** *The complex inner product is defined as*

$$(\psi, \phi) \mapsto \langle\psi| \cdot |\phi\rangle ,$$

*which is the product of the two vectors. Henceforth, we use the short notation* $\langle\psi|\phi\rangle = \langle\psi| \cdot |\phi\rangle$*. Notice that it is not commutative. Indeed, for* $\psi = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{2^n} \end{pmatrix} \in \mathbb{C}^{2^n}$ *and* $\phi = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_{2^n} \end{pmatrix} \in \mathbb{C}^{2^n}$*, we have:*

$$\begin{aligned} \langle\phi|\psi\rangle &= \begin{pmatrix} \bar{\phi_1} & \dots & \bar{\phi_{2^n}} \end{pmatrix} \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{2^n} \end{pmatrix} \\ &= \sum_{i=1}^{2^n} \bar{\phi_i}\psi_i \\ &\neq \sum_{i=1}^{2^n} \phi_i\bar{\psi_i} = \langle\psi|\phi\rangle . \end{aligned}$$

## 2.2 Gate-based quantum computing

This section aims at providing the basic notions of gate-based quantum computing necessary for the understanding of the quantum resolution of combinatorial problems.

### 2.2.1 Quantum bits

Let $|0\rangle$ and $|1\rangle$ denote the basic states of our quantum computer (the counterpart of states 0 and 1 in classical computers). The first building block of quantum algorithms is the quantum bit, also called qubit.

**Definition 2.2.1** (Qubit)**.** *We define a qubit as*

$$|q\rangle = q_0 |0\rangle + q_1 |1\rangle \ , \tag{2.1}$$

*where* $(q_0, q_1) \in \mathbb{C}^2$ *is a pair of complex numbers that satisfies the normalizing condition*

$$|q_0|^2 + |q_1|^2 = 1 \, .$$

*We say that* $q_0$ *and* $q_1$ *are the* coordinates *of* $|q\rangle$ *in the basis* $(|0\rangle, |1\rangle)$*.*

It is often convenient to use the matrix representation of $|0\rangle$ and $|1\rangle$, namely

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

With this matrix representation, the qubit $|q\rangle$ defined in (2.1) is equal to

$$|q\rangle = \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} .$$

**Example 1.** *Important examples of one-qubit states are* $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ *and* $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$*. These are usually denoted as* $|+\rangle$ *and* $|-\rangle$*, respectively.*

The algorithms typically manipulate quantum states of larger dimension. Specifically, let $n$ denote the number of qubits that our quantum computing device is able to manipulate simultaneously. An $n$-qubit state is defined by $2^n$ complex numbers that satisfy the normalizing condition and represent the normal decomposition in the canonical basis.

**Definition 2.2.2** (Canonical basis)**.** *The canonical basis of an* $n$*-qubit state is the set*

$$\mathcal{CB}_n = \left( \bigotimes_{k=1}^{n} \left| i^{(k)} \right\rangle , (i^{(1)}, \ldots, i^{(n)}) \in \{0,1\}^n \right) ,$$

*where* $i^{(k)}$ *represents the state of the* $k$*-th qubit and* $\otimes$ *is the tensor product defined earlier in Section 2.1. The canonical basis is the set of all possible combinations of tensor products of* $n$ *one-qubit basis states,* $|0\rangle$ *and* $|1\rangle$*. Thus, in the matrix representation, each canonical basis state is a column vector of* $2^n$ *components with one component equal to 1 and the rest equal to 0. It directly results that the size of the canonical basis of an* $n$*-qubit state is* $2^n$*. For more readability, we omit to write the tensor product between qubits, i.e., we refer to the canonical basis as*

$$\mathcal{CB}_n = (|i\rangle , i \in \{0,1\}^n) \, .$$

The matrix representation of canonical basis states results from the definition above. We illustrate it for the canonical basis of two qubits:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} , \qquad |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} ,$$

$$|10\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \qquad |11\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

**Definition 2.2.3** ($n$-qubit state)**.** *An $n$-qubit state $|\psi\rangle$ is a normalized linear combination of the basis states in $\mathcal{CB}_n$,*

$$|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi_i \, |i\rangle \, ,$$

*where $(\psi_i)_{i\in\{0,1\}^n} \in \mathbb{C}^{2^n}$ are its coordinates, which satisfy the normalizing condition*

$$\langle\psi|\psi\rangle = \sum_{i\in\{0,1\}^n} |\psi_i|^2 = 1 \, . \tag{2.2}$$

Notice that for $n = 1$, we find the definition of a qubit (Definition 2.2.1), as expected.

**Example 2.** *For instance, $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, called $|\Phi^+\rangle$, is a two-qubit state. Its matrix representation is:*

$$\begin{aligned}
\left|\Phi^+\right\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2}} \left( \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \right) \\
&= \frac{1}{\sqrt{2}} \left( \begin{pmatrix}1\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \right) \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix}1\\0\\0\\1\end{pmatrix} .
\end{aligned}$$

**Remark 2.2.4.** *Throughout the manuscript, we use the notation $(\psi_i)_{i\in\{0,1\}^n} \in \mathbb{C}^{2^n}$ for the coordinates of $|\psi\rangle$ in basis $\mathcal{CB}_n$. Notice that it is not always the case in the quantum computing literature where different bases may be used.*

### 2.2.2 Quantum gates

In the model of gate-based quantum computation, qubits are manipulated with quantum gates. Mathematically speaking, these quantum gates are modeled by unitary matrices. More precisely, a quantum gate that manipulates $n$-qubit states is a matrix in $\mathcal{M}_{2^n}(\mathbb{C})$ that modifies the $2^n$ complex coefficients of a quantum state such that they still satisfy the normalizing condition (2.2).

**Definition 2.2.5** (Unitary matrix)**.** *A matrix $U \in \mathcal{M}_{2^n}(\mathbb{C})$ is unitary if its inverse is equal to its conjugate transpose (denoted by the dagger symbol $\dagger$). Specifically,*

$$UU^\dagger = U^\dagger U = I \, .$$

*Notice that the application of a quantum gate is logically reversible.*

We easily see that a unitary matrix $U$ preserves the normalizing condition since, denoting $|\psi'\rangle = U|\psi\rangle$, we have

$$\langle\psi'|\psi'\rangle = \langle\psi| U^\dagger U |\psi\rangle = \langle\psi|\psi\rangle \ .$$

As an illustration, we describe below all the possible one-qubit gates, i.e., all the unitary matrices in $\mathcal{M}_2(\mathbb{C})$.

**Example 3** (Generic one-qubit gate)**.** *A generic one-qubit gate* $U = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathcal{M}_2(\mathbb{C})$ *satisfies*

$$\begin{cases} |a|^2 + |b|^2 = 1 \\ a\bar{c} + b\bar{d} = 0 \\ \bar{a}c + \bar{b}d = 0 \\ |c|^2 + |d|^2 = 1 \end{cases}$$

*Its application on a qubit* $|q\rangle = q_0|0\rangle + q_1|1\rangle$ *is:*

$$U|q\rangle = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} = \begin{pmatrix} aq_0 + bq_1 \\ cq_0 + dq_1 \end{pmatrix} = (aq_0 + bq_1)|0\rangle + (cq_0 + dq_1)|1\rangle = |q'\rangle \tag{2.3}$$

We introduce next the circuit representation of a quantum gate, which is a useful formalism to represent unitary matrices. Specifically, Figure 2.1 represents the application of the quantum gate $U$ to the one-qubit state $|q\rangle$, as in (2.3), by:

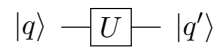


Figure 2.1: Circuit of gate $U$ applying to $|q\rangle$.

This representation easily generalizes to $n$-qubit quantum gates, where each *horizontal line* is associated with a qubit.

### 2.2.3 Quantum circuits

We can manipulate quantum gates to build other quantum gates in two different ways. The first one is the *composition*, and the other one is the *tensor product*.

**Definition 2.2.6** (Composition of quantum gates)**.** *The composition of gates only operates between gates acting on the same qubits. Let* $k$ *be the number of involved qubits. The composition of* $U_1 \in \mathcal{M}_{2^k}(\mathbb{C})$ *and* $U_2 \in \mathcal{M}_{2^k}(\mathbb{C})$ *consists of the application of* $U_1$ *followed by the application of* $U_2$*. The matrix representation of this composition is the product* $U_2U_1$*. The circuit representation of this composition is illustrated on Figure 2.2 for* $k = 3$*.*

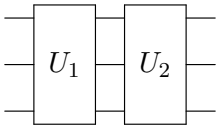


Figure 2.2: Composition of $U_1$ and $U_2$.

*It can be seen as a series sequence of gates.*

Notice that we read from right to left in the matrix representation, and from left to right in the circuit representation. Moreover, in the circuit representation, qubits are numbered in ascending order from top to bottom which is important for the tensor product circuit representation that follows.

**Definition 2.2.7** (Tensor product of quantum gates)**.** *The tensor product of gates only operates between gates acting on different qubits. Suppose $U_1 \in \mathcal{M}_{2^k}(\mathbb{C})$ applies on the first $k$ qubits and $U_2 \in \mathcal{M}_{2^{k'}}(\mathbb{C})$ applies on the $k'$ following ones. Their tensor product is the application of $U_1$ and $U_2$ on respective qubits in parallel. The matrix representation of this tensor product is $U_1 \otimes U_2$ (Definition 2.1.1). The circuit representation is depicted on Figure 2.3 for $k = 3$ and $k' = 2$.*

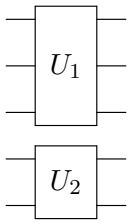


Figure 2.3: Tensor product of $U_1$ and $U_2$.

Notice that when we apply a quantum gate on $k$ qubits of an $n$-qubit system, it supposes that we apply identity gate $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ on the qubits not concerned. For instance, let us consider a 3-qubit system on which we apply $U \in \mathcal{M}_2(\mathbb{C})$ on qubit number 2. The matrix representation of the resulting 3-qubit gate is $I \otimes U \otimes I$, and its circuit representation is illustrated on Figure 2.4, where the application of $I$ is usually replaced by a simple wire.

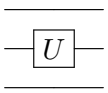


Figure 2.4: Application of $U$ to qubit number 2.

One readily verifies that both composition and tensor product transform unitary matrices into a resulting unitary matrix.

Throughout, we consider that a quantum algorithm is a quantum circuit acting on $n$ qubits, that is, a sequence of quantum gates' compositions and/or tensor products. These quantum gates can be $k$-qubit gates, for $k \in [n]$. However, quantum gates involving many qubits are typically not implementable natively on quantum computers and need to be decomposed into smaller and simpler gates. This set of small gates can be considered as the quantum counterpart of the elementary logic gates used in classical circuit computing to assess the circuit complexity of a classical algorithm. Thus, an $n$-qubit quantum algorithm is described by a unitary matrix in $\mathcal{M}_{2^n}(\mathbb{C})$, and we decompose it as a sequence of universal gates (Definition 2.2.8) to obtain the complexity of the quantum algorithm.

**Definition 2.2.8** (Set of universal gates)**.** *A set of quantum gates PU is* universal *if we can decompose any $n$-qubit quantum gate through a circuit composed solely of the gates in PU.*

Fortunately, there exist different universal sets of quantum gates. We introduce below such a set formed of four types of gates. First, we consider the three following families of one-qubit gates, each of which is parametrized by a real number $\theta \in \mathbb{R}$:

$$R_X(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} , \tag{2.4}$$

$$R_Y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} , \tag{2.5}$$

$$R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} . \tag{2.6}$$

Notice that these gates are often referred to as rotation gates because they correspond to rotations in a certain representation of qubits, known as the Bloch sphere (Mosseri and Dandoloff, 2001). Second, we consider the two-qubit gate $CX$, often called the "Control-NOT gate":

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{2.7}$$

$CX$ applies gate $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, which is the equivalent of the NOT gate in quantum computing, on the second qubit if and only if the first qubit is in state $|1\rangle$. Hence the name "Control-NOT". Indeed, for instance, we have

$$CX\,|01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle ,$$

and

$$CX\,|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle .$$

In other words, we can define $CX$ on the canonical basis as

$$CX\,|a, b\rangle = |a, b \oplus a\rangle ,$$

for $a, b \in \{0, 1\}$, where $\oplus$ is the addition modulo 2.

**Remark 2.2.9** (Notation on gates indexes)**.** *Henceforth, we use the notation $R_{X,i}$ for the application of $R_X$ on qubit $i$ (and the application of identity matrix on the remaining qubits). We do the same with $R_{Y,i}$ and $R_{Z,i}$. We note $CX_{i,j}$ the application of $CX$ gate to qubits $i$ and $j$: $X$ is applied to qubit $j$ if and only if qubit $i$ is in state $|1\rangle$.*

**Theorem 2.2.10** (Universal gates (Nielsen and Chuang, 2010))**.** *The set of one-qubit gates and the $CX$ gate (2.7) is universal. Thus, because any one-qubit gate is the composition of rotation gates (2.4)–(2.6), the set $PU = \{R_X(\alpha), R_Y(\beta), R_Z(\gamma), CX : \alpha, \beta, \gamma \in \mathbb{R}\}$ is universal.*

In comparison, the sets of classical logic gates {NAND}, {NOR}, {NOT, AND} and {NOT, OR} are universal for classical computation. Indeed, we can compute any arbitrary classical function with them. In view of the above, we typically consider that the quantum counterpart of the classical number of elementary operations is the number of universal gates used to decompose the circuit. Of course, this decomposition depends on the set of universal gates $PU$ considered, but the number of gates required is the same with any set, modulo a multiplicative constant (Nielsen and Chuang, 2010). Thereafter, we only consider the set $PU$ defined in Theorem 2.2.10. This choice is particularly motivated by the algorithms we study in Chapter 3, as it will be convenient to express them with this set.

**Definition 2.2.11** (Complexity of quantum circuits)**.** *Let us consider a family of quantum circuits $(\mathcal{Q}_n)_{n\in\mathbb{N}}$. We say that the gate complexity of this family is $\mathcal{O}(C(n))$ if $\mathcal{Q}_n$ is a circuit that applies on $n$ qubits and contains $f(n)$ universal quantum gates, where $f(n) = \mathcal{O}(C(n))$.*

In other words, if we consider the family of circuits $(\mathcal{Q}_n)_{n\in\mathbb{N}}$ where $\mathcal{Q}_n$ is a circuit on $n$ qubits and is decomposed on $\mathcal{O}(poly(n))$ universal quantum gates, then this family is said to be *efficient*.

**Observation 2.2.12.** *Let $U_1$ and $U_2$ be two quantum circuits, with complexity $\mathcal{O}(C_1(n))$ and $\mathcal{O}(C_2(n))$, respectively. The complexity of the composition $U_1 \cdot U_2$ is*

$$\mathcal{O}(C_1(n) + C_2(n)) = \mathcal{O}(\max(C_1(n), C_2(n)))\,.$$

*The tensor product $U_1 \otimes U_2$ has the same complexity.*

### 2.2.4 Non-classical behaviors

The quantum algorithms rely on three characteristics of quantum states with no classical equivalent: measurement, superposition, and entanglement. Let us present these notions through the bare minimum mathematical background.

#### 2.2.4.1 Measurement

We need to measure a quantum state $|\psi\rangle$ to get information from it. Otherwise, no information is accessible. The peculiar property of measurement is that it only extracts partial information from the quantum state: the single measurement output of $|\psi\rangle$ is a bitstring.

**Definition 2.2.13** (Measurement)**.** *In the gate-based quantum model, the measurement $\mathcal{M}$ of an $n$-qubit state $|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi_i |i\rangle$ outputs the $n$-bitstring $i$ with probability $|\psi_i|^2$. After having been measured, state $|\psi\rangle$ no longer exists: it has been replaced by the state $|i\rangle$.*

For example, measuring qubit $|q\rangle = q_0|0\rangle + q_1|1\rangle$ outputs 0 with probability $|q_0|^2$ and 1 with probability $|q_1|^2$, and changes the state $|q\rangle$ to $|0\rangle$ and $|1\rangle$, respectively. A measurement appears as a loss of information. Indeed, we describe an $n$-qubit state by $2^n$ normalized complex coefficients, but we only extract an $n$-bitstring after measuring it. The perfect knowledge of the probabilities representing a given quantum state $|\psi\rangle$, namely the square module of each of its coordinates $(|\psi_i|^2)_{i\in\{0,1\}^n} \in [0,1]^{2^n}$, can be obtained only if we measure $|\psi\rangle$ an infinite number of times. Notice that it requires resetting state $|\psi\rangle$ after each measurement.

**Remark 2.2.14** (Sampling of quantum states)**.** *In reality we are limited to approximating a given quantum state through sampling. In particular, if $|\psi\rangle$ is the result of an algorithm, this means we have to repeat the same algorithm for every measurement of $|\psi\rangle$ we wish to perform.*

#### 2.2.4.2 Superposition

Classically, the state of an $n$-bit computer is given by a bitstring in $\{0,1\}^n$. We have seen so far that the state of an $n$-qubit quantum computer is given by its coordinates $(\psi_i)_{i\in\{0,1\}^n} \in \mathbb{C}^{2^n}$, which satisfy $\sum_{i\in\{0,1\}^n} |\psi_i|^2 = 1$. In general, more than one of the coordinates are different from 0, meaning that measuring $|\psi\rangle$ may result in different bitstring $i \in \{0,1\}^n$.

**Definition 2.2.15** (Superposition)**.** *A quantum state $|\psi\rangle$ is said in superposition if $|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi_i |i\rangle$ where there are at least two terms with non-zero coefficients in the sum. A quantum state that is not a basis state is in superposition.*

The following Hadamard gate is the usual one-qubit gate that produces superposition starting from a canonical basis state.

**Example 4** (Hadamard gate)**.** *The Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is essential in quantum computing because it creates superposition starting from a basis state. We obtain the state $|+\rangle$, respectively $|-\rangle$, of Example 1 by applying $H$ on $|0\rangle$, respectively $|1\rangle$:*

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \ ,$$
$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \ .$$

**Example 5** (Uniform superposition)**.** *The two states $|+\rangle$ and $|-\rangle$ are in uniform superposition because both have the same probability of being measured as 0 or 1. In general, an $n$-qubit state uniformly superposed is equal to $\frac{1}{\sqrt{2^n}} \sum_{j\in\{0,1\}^n} e^{i\alpha_j} |j\rangle$, with $\alpha_j \in [0, 2\pi[$, $\forall j \in \{0,1\}^n$. In what follows, we shall often use the uniformly superposed $n$-qubit state $|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} |i\rangle$.*

Notice that applying an $n$-qubit quantum gate $U$ to $|\psi\rangle$ possibly modifies the $2^n$ coordinates of $|\psi\rangle$ since

$$U|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi'_i |i\rangle \ ,$$

where possibly each $\psi'_i$ differs from $\psi_i$. This is the case, for instance, when applying the tensor product of $n$ Hadamard gates, each applied to a qubit initially in state $|0\rangle$, specifically

$$H^{\otimes n}|0\rangle^{\otimes n} = \bigotimes_{i=1}^{n} H|0\rangle = \bigotimes_{i=1}^{n} \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} |i\rangle = |+\rangle^{\otimes n} \ . \tag{2.8}$$

Equation (2.8) illustrates the potential benefit of quantum circuits: applying $\mathcal{O}(n)$ universal one-qubit gates impacts the exponentially many coefficients of $|\psi\rangle$. Indeed, one readily verifies that $H = R_X(\pi)R_Y(\frac{\pi}{2})$ modulo a global phase[1], so $H^{\otimes n}$ amounts to applying $2n$ universal one-qubit gates.

[1]Two quantum states $|\psi\rangle$ and $|\psi'\rangle = e^{i\alpha}|\psi\rangle$, with $\alpha \in [0, 2\pi[$, that only differ by a global phase are indiscernible by measurement. Thus, we do not consider global phase of quantum states nor quantum gates.

#### 2.2.4.3 Entanglement

Each quantum state is either a product state or an entangled state. Entanglement has the peculiar and helpful property that we can apply a circuit only on a part of the $n$-qubit system, and as a result, the whole system is affected.

**Definition 2.2.16** (Product state)**.** *An $n$-qubit state is a product state if it is the tensor product of $n$ one-qubit states. In other words, an $n$-qubit state $|\psi\rangle$ is a product state if it exists $2n$ complex coefficients $(q_0^{(j)}, q_1^{(j)})_{j\in[n]}$ such that*

$$|\psi\rangle = \bigotimes_{j=1}^{n}(q_0^{(j)}\,|0\rangle + q_1^{(j)}\,|1\rangle),\ \textit{with}\ |q_0^{(j)}|^2 + |q_1^{(j)}|^2 = 1, \forall j \in [n]\,.$$

*Thus, each state of a qubit that composes $|\psi\rangle$ can be described independently of the states of the other.*

If an $n$-qubit state is not a product state, it is an entangled state.

**Definition 2.2.17** (Entangled state)**.** *An $n$-qubit state $|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi_i\,|i\rangle$ is entangled if the numerical values of its coordinates $(\psi_i)_{i\in\{0,1\}^n} \in \mathbb{C}^{2^n}$ admit no solution $(q_0^{(j)}, q_1^{(j)})_{j\in[n]} \in \mathbb{C}^{2n}$ to the system*

$$\begin{cases} \displaystyle\sum_{i\in\{0,1\}^n} \psi_i\,|i\rangle = \bigotimes_{j=1}^{n}(q_0^{(j)}\,|0\rangle + q_1^{(j)}\,|1\rangle) \\ |q_0^{(j)}|^2 + |q_1^{(j)}|^2 = 1, \forall j \in [\![1, n]\!] \end{cases}$$

*It means that operations performed on some coordinates of the entangled state can affect the other coordinates without direct operations on them.*

Notice that when an $n$-qubit system is entangled, it makes no sense to speak about qubit number $k \in [n]$ because isolated qubits are not defined. Notice also that there is a difference between superposition and entanglement. A state is in superposition if at least two non-zero coefficients are in its basis decomposition. A state is entangled if it cannot be written as a tensor product of independent qubits, implying that it is in superposition.

To illustrate the notion of entanglement, we consider the following quantum circuit on Figure 2.5. Starting from a two-qubit product state $|00\rangle$, it results an entangled state.

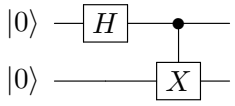


Figure 2.5: Entangling circuit.

This circuit is the application of Hadamard gate on qubit 1, followed by $CX_{1,2}$. The quantum

state that results is $|\Phi^+\rangle$ of Example 2:

$$\begin{aligned} CX_{1,2}(H \otimes I)\,|00\rangle &= CX_{1,2}\left(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\right) \\ &= \frac{1}{\sqrt{2}}\left(CX_{1,2}\,|00\rangle + CX_{1,2}\,|10\rangle\right) \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ &= \left|\Phi^+\right\rangle . \end{aligned}$$

We prove now by contradiction that $|\Phi^+\rangle$ is entangled. Suppose that $|\Phi^+\rangle$ is a product state. Thus, there exist $(q_0^{(0)}, q_1^{(0)}) \in \mathbb{C}^2$ such as $|q_0^{(0)}|^2 + |q_1^{(0)}|^2 = 1$ and $(q_0^{(1)}, q_1^{(1)}) \in \mathbb{C}^2$ such as $|q_0^{(1)}|^2 + |q_1^{(1)}|^2 = 1$, satisfying:

$$\begin{aligned} \left|\Phi^+\right\rangle &= (q_0^{(0)}\,|0\rangle + q_1^{(0)}\,|1\rangle) \otimes (q_0^{(1)}\,|0\rangle + q_1^{(1)}\,|1\rangle) \\ &= q_0^{(0)}q_0^{(1)}\,|00\rangle + q_0^{(0)}q_1^{(1)}\,|01\rangle + q_1^{(0)}q_0^{(1)}\,|10\rangle + q_1^{(0)}q_1^{(1)}\,|11\rangle . \end{aligned}$$

Because the decomposition of a quantum state's coordinates is unique in the canonical basis, it follows by identification:

$$\begin{cases} q_0^{(0)}q_0^{(1)} &= \frac{1}{\sqrt{2}} \\ q_0^{(0)}q_1^{(1)} &= 0 \\ q_1^{(0)}q_0^{(1)} &= 0 \\ q_1^{(0)}q_1^{(1)} &= \frac{1}{\sqrt{2}} \end{cases}$$

This equation system admits no solution. We deduce that $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is not a product state, hence it is entangled.

## 2.3 Quantum circuit complexity

In computer science, the complexity of an algorithm represents a measure of its performance. We will require such measure to assess the performance of algorithms, specifically in Chapters 5, 6 and 7. As in classical computing, the complexity of a quantum algorithm (i.e. a quantum circuit in the gate-based model) can be measured in different ways. For instance, there is the *space complexity* referring to the number of qubits on which the quantum circuit applies. There is also the *query complexity*, which is defined by the number of calls done to the *oracle*, where oracle means access to the input data. For example, Grover Search (Grover, 1996) makes $\mathcal{O}(\sqrt{2^n})$ queries to the reversible oracle representing the function $f : \{0,1\}^n \to \{0,1\}$ to find with high probability an element $x$ such that $f(x) = 1$. Notice that in practice, the oracle can be a quantum circuit generated independently by another process, or often represents access to quantum memory. In this thesis, we assess the performance of quantum algorithms with the *gate complexity* which is defined by the number of universal gates (see Definition 2.2.8).

Henceforth, we use the following usual asymptotic notations for the gate complexity. Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be two functions. We say that

$$f(n) = \mathcal{O}(g(n))$$

if there exists $M > 0$ and $n_0 \in \mathbb{N}$ such that, $\forall n \geq n_0$,

$$f(n) \leq Mg(n)\,.$$

In other words, for sufficiently large values of $n$, the function $f$ grows no faster than a constant (positive) multiple of $g$. We denote by $\mathcal{O}^*$ the same as for $\mathcal{O}$ but ignoring polynomial factors. For instance, for $f : n \mapsto 7n^2 3^n$, we have $f(n) = \mathcal{O}^*(3^n)$. Similarly, we refer to $\tilde{\mathcal{O}}$ as $\mathcal{O}$ but we ignore the polylogarithmic factors. For example, for $f : n \mapsto \log(n)^4 n^5$, we have $f(n) = \mathcal{O}^*(n^5)$.

Eventually, we mention an observation regarding the complexity of a classical algorithm executed as a quantum circuit.

**Observation 2.3.1** (Classical algorithm into quantum circuit (Bennett, 1973))**.** *Any classical algorithm $\mathcal{A}$ can be executed as a quantum circuit $U_{\mathcal{A}}$ by preserving the number of gates but increasing the size memory. This additional cost comes from the fact that $U_{\mathcal{A}}$ must be reversible. Specifically, if $\mathcal{A}$ uses $T$ gates and $S$ bits of memory, $U_{\mathcal{A}}$ uses $\mathcal{O}(T)$ gates and $\mathcal{O}(T + S)$ bits of memory.*

# 3

# Variational Quantum Algorithms

Today, the quantum algorithms used for solving optimization problems and conducting preliminary numerical tests are the Variational Quantum Algorithms (VQAs). VQAs are heuristics that alternate between a quantum circuit and a classical optimizer. They tackle unconstrained optimization problems with binary variables. Even if these algorithms have no performance guarantee for general problems, they are of great interest today because they have the convenient property of an adjustable quantum circuits' depth, making them implementable on the current NISQ computers. The variational approach of VQAs consists of probing the initial search space with a relatively small set of parameters optimized classically. Specifically, these parameters describe a probability distribution over the search space. Thus, VQAs take advantage of the quantum computing principle that generates a probability distribution on an exponential space in a short sequence of quantum gates.

In this chapter, we provide a mathematical description of Variational Quantum Algorithms (VQAs) and focus on one of them, the Quantum Approximate Optimization Algorithm (QAOA) of Farhi et al. (2014). Indeed, this is the first variational algorithm that tackles combinatorial optimization problems, such as the MAX-CUT problem. We shall also devote particular attention to unconstrained problems for which the objective function is polynomial. More precisely, we begin by defining the different parts that constitute and define a VQA, namely the quantum circuit, the classical optimizer, and the guiding function driving the classical optimization. We characterize each part with properties that could enable theoretical guarantees. Afterward, we study QAOA, describing the necessary reformulation of the initial unconstrained problem into a Hermitian matrix and analyzing this algorithm in light of the previous properties. We also provide a universal decomposition of QAOA's quantum circuit for the general case where the objective function is polynomial, and we give a condensed overview of empirical trends and theoretical limitations of QAOA. Eventually, we propose a generic method to reformulate constrained problems with integer variables, polynomial objective function, and polynomial constraints into an eligible problem for a resolution with VQAs: an unconstrained problem with binary variables and polynomial objective function.

## 3.1 Unconstrained optimization

In this section, we present the class of Variational Quantum Algorithms (VQAs) (Cerezo et al., 2021) from the perspective of solving combinatorial optimization problems. VQAs are studied today because they represent an alternative approach that reduces the quality and quantity of the quantum resources needed (Peruzzo et al., 2014): they are hybrid algorithms, requiring both quantum and classical computations. Specifically, they are designed to run on the NISQ era (Preskill, 2018) where quantum computers are noisy with few qubits, enabling them to harness low-depth quantum circuits.

Throughout this section, we consider optimization problems of the form

$$\min_{x\in\{0,1\}^n} f(x)\,, \tag{3.1}$$

where $f$ is any function defined on $\{0,1\}^n$. We note $\mathcal{F}$ the set of optimal solutions.

### 3.1.1 General description

Variational Quantum Algorithms (VQAs) are hybrid algorithms that, given an input $|0\rangle^{\otimes n}$, alternate between a quantum and a classical part. Henceforth, we note $|0_n\rangle$ the state $|0\rangle^{\otimes n}$ to ease the reading. Let us first provide a high-level description of the key elements of VQAs that are detailed in this section. Let $d \in \mathbb{N}$, the three key elements of VQAs are:

- a *parametrized quantum circuit*, $U : \mathbb{R}^d \to \mathcal{M}_{2^n}(\mathbb{C})\,$,
- a *guiding function*, $g : \mathbb{R}^d \to \mathbb{R}\,$,
- and a *classical optimizer*, which is an algorithm $\mathcal{A}$ that optimizes $g$ over space $\mathbb{R}^d\,$.

These elements, the parametrized quantum circuit and the two others, are detailed and illustrated in Subsections 3.1.2 and 3.1.3, respectively. Before that, we provide a general overview of VQAs in Algorithm 1.

The main idea of a VQA is as follows. The main loop of the algorithm executes the following steps until the classical optimizer stops according to a given stopping criterion. First, given $\theta \in \mathbb{R}^d$, the *quantum part* executes the parametrized quantum circuit $U(\theta)$ for several times to sample the quantum state (see Remark 2.2.14). It results in a distribution probability over $\{0,1\}^n$ that we note $\{p_\theta(x) : x \in \{0,1\}^n\}$. Second, the *classical part* computes the cost of this state through the evaluation of the guiding function according to the sampling results, specifically, $g(\theta) = G(p_\theta, f)$ for a given function $G$ that is constructed from $p_\theta$ and $f$. More precisely, for a given $\theta$, the computation of $G$ requires only the values $f(x)$ for $x$ such that $p_\theta(x) > 0$. Eventually, this cost value is given to the classical optimizer $\mathcal{A}$, which outputs a new parameter in order to minimize $g$. Notice that, between the two parts, the best-found solution is possibly updated by a classical computer.

In the following subsections, we detail each part of VQAs and show how specific choices of $g$ and $U$ ensure that VQAs optimize $f$. Let us begin with the quantum part.

### 3.1.2 Quantum part

The quantum part of VQAs applies a quantum circuit on the $n$-qubit system that constitutes the quantum computer. Importantly, *variational* in VQAs stands for the parametrization of the quantum circuit. Let $d \in \mathbb{N}$ be the number of parameters.

**Definition 3.1.1.** *A parametrized quantum circuit is a continuous function $U : \mathbb{R}^d \to \mathcal{M}_{2^n}(\mathbb{C})$ mapping any $\theta \in \mathbb{R}^d$ to unitary matrix $U(\theta)$.*

As defined in Subsection 2.2.3, a quantum circuit is a sequence of universal quantum gates' compositions and/or tensor products. Thus, all coefficients of matrix $U(\theta)$ are continuous functions on $\mathbb{R}^d$.

**Example 6.** *A simple example of a parametrized quantum circuit for $n = 3$ and $d = 3$ is depicted in Figure 3.1.*

**Algorithm 1:** Variational Quantum Algorithm

**Input:** guiding function $g$, parametrized quantum circuit $U$, classical optimizer $\mathcal{A}$, initial parameter $\theta_0$

**Output:** approximate minimum $f^*$ and the corresponding minimizer $x^*$

```
θ ← θ_0;
f* ← +∞;
x* ← 0^n;
while stopping criterion of the classical optimizer is false do
    begin quantum part
        for size of the sampling do
            measure the state U(θ)|0⟩^{⊗n}, outputting x for some x ∈ {0,1}^n;
        let p_θ(x) be the frequency of x ∈ {0,1}^n in the above sampling;
    begin solution update
        for x such that p_θ(x) > 0 do
            compute f(x);
            if f(x) < f* then
                f* ← f(x);
                x* ← x;
    begin classical part
        compute g(θ) = G(p_θ, f);
        given g(θ) (and possibly its derivatives), A outputs θ';
        θ ← θ';
```

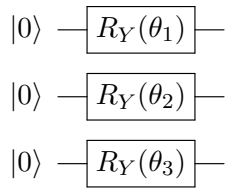


Figure 3.1: Quantum circuit parametrized by $(\theta_1, \theta_2, \theta_3) \in \mathbb{R}^3$.

*The expression of $U(\theta)$ for this circuit is as follows:* $\forall \theta = (\theta_1, \theta_2, \theta_3) \in \mathbb{R}^3$,

$$\begin{aligned} U(\theta) &= R_Y(\theta_1) \otimes R_Y(\theta_2) \otimes R_Y(\theta_3) \\ &= \begin{pmatrix} \cos\frac{\theta_1}{2} & -\sin\frac{\theta_1}{2} \\ \sin\frac{\theta_1}{2} & \cos\frac{\theta_1}{2} \end{pmatrix} \otimes \begin{pmatrix} \cos\frac{\theta_2}{2} & -\sin\frac{\theta_2}{2} \\ \sin\frac{\theta_2}{2} & \cos\frac{\theta_2}{2} \end{pmatrix} \otimes \begin{pmatrix} \cos\frac{\theta_3}{2} & -\sin\frac{\theta_3}{2} \\ \sin\frac{\theta_3}{2} & \cos\frac{\theta_3}{2} \end{pmatrix} \end{aligned}$$

$$= \begin{pmatrix} c_1c_2c_3 & -c_1c_2s_3 & -c_1s_2c_3 & c_1s_2s_3 & -s_1c_2c_3 & s_1c_2s_3 & s_1s_2c_3 & -s_1s_2s_3 \\ c_1c_2s_3 & c_1c_2c_3 & -c_1s_2s_3 & -c_1s_2c_3 & -s_1c_2s_3 & -s_1c_2c_3 & s_1s_2s_3 & s_1s_2c_3 \\ c_1s_2c_3 & -c_1s_2s_3 & c_1c_2c_3 & -c_1c_2s_3 & -s_1s_2c_3 & s_1s_2s_3 & -s_1c_2c_3 & s_1c_2s_3 \\ c_1s_2s_3 & c_1s_2c_3 & c_1c_2s_3 & c_1c_2c_3 & -s_1s_2s_3 & -s_1s_2c_3 & -s_1c_2s_3 & -s_1c_2c_3 \\ s_1c_2c_3 & -s_1c_2s_3 & -s_1s_2c_3 & s_1s_2s_3 & c_1c_2c_3 & -c_1c_2s_3 & -c_1s_2c_3 & c_1s_2s_3 \\ s_1c_2s_3 & s_1c_2c_3 & -s_1s_2s_3 & -s_1s_2c_3 & c_1c_2s_3 & c_1c_2c_3 & -c_1s_2s_3 & -c_1s_2c_3 \\ s_1s_2c_3 & -s_1s_2s_3 & s_1c_2c_3 & -s_1c_2s_3 & c_1s_2c_3 & -c_1s_2s_3 & c_1c_2c_3 & -c_1c_2s_3 \\ s_1s_2s_3 & s_1s_2c_3 & s_1c_2s_3 & s_1c_2c_3 & c_1s_2s_3 & c_1s_2c_3 & c_1c_2s_3 & c_1c_2c_3 \end{pmatrix},$$

*where* $c_i = \cos\frac{\theta_i}{2}$ *and* $s_i = \sin\frac{\theta_i}{2}$ *for* $i \in [3]$.

**Remark 3.1.2.** *The use of the generalized circuit to n qubits* $\bigotimes_{i=1}^{n} R_{Y,i}(\theta_i)$ *amounts to a continuous relaxation of the* $\{0,1\}$*-problem* (3.1)*, where each decision variable* $x_i \in [0,1]$ *is represented by a rotation angle* $\theta_i \in [0, 2\pi]$ *as follows:*

$$x_i = \left(\cos\frac{\theta_i}{2}\right)^2.$$

### 3.1.3 Classical part

The classical part of VQAs consists of a classical optimization over the parameters $\theta \in \mathbb{R}^d$. The classical optimizer essentially aims at finding the optimal parameters $\theta^*$ that lead to optimal solutions of the initial problem (3.1) with high probability, specifically, such that

$$\sum_{s\in\mathcal{F}} |\langle s| U(\theta^*) |0_n\rangle|^2 \geq 1 - \epsilon\,, \tag{3.2}$$

for small $\epsilon > 0$. Henceforth, we use notation $|x\rangle$ instead of $|i\rangle$ to efficiently recall that we deal with solutions of optimization problems.

The classical part is characterized by two aspects: the function that guides the optimization and the optimizer itself.

#### 3.1.3.1 Guiding function

Let $g : \mathbb{R}^d \to \mathbb{R}$ be the guiding function, formally defined in Definition 3.1.3 below, that the classical optimizer minimizes. The function $g$ acts as a link between the quantum and classical parts. For a given $\theta \in \mathbb{R}^d$, we evaluate $U(\theta)|0_n\rangle$ *according to* $f$ as we will exemplify below. Notice that $f$ and $g$ are distinct since $g$ is defined on $\mathbb{R}^d$ and outputs a quality measure of an $n$-qubit quantum state whereas $f$ is defined on $\{0,1\}^n$. Let us denote $\mathcal{F}_{\text{quant}} = \left\{\sum_{s\in\mathcal{F}} \psi_s |s\rangle : \sum_{s\in\mathcal{F}} |\psi_s|^2 = 1\right\}$ as the set of quantum states that are superpositions of optimal solutions of problem (3.1). Naturally, we would like to define $g$ such that minimizing $g$ tends to minimize $f$.

**Definition 3.1.3** (Guiding function)**.** *Let* $g : \mathbb{R}^d \to \mathbb{R}$ *be a function and* $\mathcal{G}$ *be its set of minimizers. We call* $g$ *a guiding function for* $f$ *with respect to* $U$ *if* $g$ *is continuous and*

$$\{U(\theta)|0_n\rangle : \theta \in \mathcal{G}\} \subseteq \mathcal{F}_{quant}\,. \tag{3.3}$$

In other words, optima of a guiding function $g$ must lead to optima of $f$ or superpositions of optima of $f$. Indeed, Equation (3.3) implies that measuring the quantum state $U(\theta^*)|0\rangle^{\otimes n}$

for $\theta^* \in \mathcal{G}$ outputs with probability 1 an optimal solution $s \in \mathcal{F}$ of the initial problem. Thus, minimizing $g$ amounts to minimizing $f$, and finding $\mathcal{F}$ is done by finding $\mathcal{G}$. The latter is found with the classical optimizer. Without any information on $\mathcal{F}$, we need to choose a quantum circuit $U$ such that any optimal solution $s \in \mathcal{F}$ is reachable, specifically,

$$\{U(\theta)\,|0_n\rangle : \theta \in \mathbb{R}^d\} \supseteq \mathcal{CB}_n\,. \tag{3.4}$$

Notice that this condition is weak and easily satisfied. For instance, the circuit depicted in Figure 3.1 satisfies this condition. If ever one is interested in finding all optimal solutions of problem (3.1), the circuit and the guiding function should satisfy instead the stronger condition

$$\{U(\theta)\,|0_n\rangle : \theta \in \mathcal{G}\} = \mathcal{F}_{\text{quant}}\,. \tag{3.5}$$

In that case, $U$ satisfying (3.4) is not enough. Without any information on $\mathcal{F}$, we need to choose $U$ that can reach any $n$-qubit quantum states, specifically,

$$\{U(\theta)\,|0_n\rangle : \theta \in \mathbb{R}^d\} = \left\{ \sum_{x\in\{0,1\}^n} \psi_x\,|x\rangle : \sum_{x\in\{0,1\}^n} |\psi_x|^2 = 1 \right\}.$$

A popular choice for the guiding function in the literature is the mean function

$$g_{\text{mean}}(\theta) = \sum_{x\in\{0,1\}^n} p_\theta(x) f(x)\,, \tag{3.6}$$

where $p_\theta(x) = |\,\langle x|\,U(\theta)\,|0_n\rangle\,|^2$ is the probability of finding $x$ when $U(\theta)\,|0_n\rangle$ is measured. We show next that $g_{\text{mean}}$ is indeed a guiding function according to Definition 3.1.3.

**Proposition 3.1.4.** *Function $g_{mean}$ is a guiding function.*

*Proof.* Let us prove that $g_{\text{mean}}$ is continuous. Let $x \in \{0,1\}^n$. The function $\theta \mapsto p_\theta(x) = |\,\langle x|\,U(\theta)\,|0_n\rangle\,|^2$ is continuous, because each coefficient of $U(\theta)$ is continuous. Thus, because multiplication and addition preserve continuity, $g_{\text{mean}}$ is continuous.

We prove by contradiction that (3.3) holds. Let $\theta \in \mathcal{G}$ and let us consider the quantum state $|\psi(\theta)\rangle = U(\theta)\,|0_n\rangle$. We write its decomposition in the canonical basis as follows: $|\psi(\theta)\rangle = \sum_{x\in\{0,1\}^n} \psi_x\,|x\rangle$.

Assume that $|\psi(\theta)\rangle \notin \mathcal{F}_{\text{quant}}$. By definition, there exists $x_0 \in \{0,1\}^n$ such that $x_0 \notin \mathcal{F}$ and $|\psi_{x_0}| \neq 0$. Thus,

$$\begin{aligned} g_{\text{mean}}(\theta) &= \sum_{x\in\{0,1\}^n} |\psi_x|^2 f(x) \\ &= \sum_{x\in\mathcal{F}} |\psi_x|^2 f(x) + \sum_{x\notin\mathcal{F}} |\psi_x|^2 f(x) \\ &= \left(\sum_{x\in\mathcal{F}} |\psi_x|^2\right) f^* + \sum_{x\notin\mathcal{F}} |\psi_x|^2 f(x)\,, \end{aligned}$$

where $f^*$ is the optimal value of $f$, reached on $\mathcal{F}$. By definition, $f(x) > f^*, \forall x \notin \mathcal{F}$, and because we assume that $|\psi_{x_0}| \neq 0$, thus the second term of the sum is bounded below as follows:

$\sum_{x\notin\mathcal{F}} |\psi_x|^2 f(x) > \left(\sum_{x\notin\mathcal{F}} |\psi_x|^2\right) f^*$, where $\sum_{x\notin\mathcal{F}} |\psi_x|^2 \geq |\psi_{x_0}|^2 > 0$. Thus,

$$g_{\text{mean}}(\theta) > \left(\sum_{x\in\mathcal{F}} |\psi_x|^2\right) f^* + \left(\sum_{x\notin\mathcal{F}} |\psi_x|^2\right) f^* = f^* .$$

This contradicts the previous statement that $\theta \in \mathcal{G}$, as one readily verifies that the minimum of $g_{\text{mean}}$ is $g^*_{\text{mean}} = f^*$. □

**Example 7.** *We illustrate the mean function on the 3-qubit quantum circuit $U(\theta)$ depicted in Figure 3.1. The generalization of its computation for $n$ qubits is trivial since it needs to replace 3 by $n$. The single application of rotation gate $R_Y$ (2.5) of angle $\theta_i$ on a qubit initially on state $|0\rangle$ is*

$$\begin{aligned} R_Y(\theta_i)|0\rangle &= \cos\frac{\theta_i}{2}|0\rangle + \sin\frac{\theta_i}{2}|1\rangle \\ &= \sum_{j\in\{0,1\}} \cos\frac{\theta_i - j\pi}{2}|j\rangle , \end{aligned}$$

*since $\sin(\phi) = \cos\left(\phi - \frac{\pi}{2}\right)$ for $\phi \in \mathbb{R}$. Eventually, the quantum state resulting from $U(\theta)$ is*

$$\begin{aligned} U(\theta)|0_3\rangle &= \bigotimes_{i=1}^{3} R_{Y,i}(\theta_i)|0\rangle \\ &= \sum_{j_1,j_2,j_3\in\{0,1\}} \left(\prod_{i=1}^{3} \cos\frac{\theta_i - j_i\pi}{2}\right)|j_1 j_2 j_3\rangle . \end{aligned}$$

*Thus, the probability to measure $x = (x_1, x_2, x_3) \in \{0,1\}^3$ is*

$$p_\theta(x) = \left(\prod_{i=1}^{3} \cos\frac{\theta_i - x_i\pi}{2}\right)^2 , \tag{3.7}$$

*and the expression of $g_{mean}$ of equation (3.6) directly results from it.*

Other functions are compatible with Definition 3.1.3. One of these functions encountered in the literature is the Gibbs function (Li et al., 2020) which stems from statistical mechanics. Let $\eta > 0$ be a parameter to be set. The Gibbs function is defined as

$$g_{G,\eta}(\theta) = -\ln\left(\sum_{x\in\{0,1\}^n} p_\theta(x) e^{-\eta f(x)}\right) . \tag{3.8}$$

The choice of this function is motivated by the exponential shape that highly rewards the increase of probabilities of low-cost states. Notice that for small $\eta$, minimizing the Gibbs function is essentially equivalent to minimizing the mean function in the sense that the Taylor series of $g_{G,\eta}$ at first order in $\eta = 0$ gives $g_{G,\eta} = \eta g_{\text{mean}}$. We show next that $g_{G,\eta}$ is indeed a guiding function according to Definition 3.1.3.

**Proposition 3.1.5.** *Let $\eta > 0$. Function $g_{G,\eta}$ is a guiding function.*

*Proof.* Let us prove that $g_{G,\eta}$ is continuous. Let $x \in \{0,1\}^n$. The function $\theta \mapsto p_\theta(x) = |\langle x| U(\theta)|0_n\rangle|^2$ is continuous, because each coefficient of $U(\theta)$ is continuous. Thus, because

multiplication, addition, and composition preserve continuity, $g_{G,\eta}$ is continuous.

Let $\eta > 0$. Let us prove by contradiction that (3.3) holds. As before, let $\theta \in \mathcal{G}$ and let us consider the quantum state $|\psi(\theta)\rangle = U(\theta)\,|0_n\rangle$. We write $|\psi(\theta)\rangle = \sum_{x\in\{0,1\}^n} \psi_x\,|x\rangle$ its decomposition in the canonical basis. Assume that $|\psi(\theta)\rangle \notin \mathcal{F}_{\text{quant}}$. Thus, there exists $x_0 \in \{0,1\}^n$ such that $x_0 \notin \mathcal{F}$ and $|\psi_{x_0}| \neq 0$. Thus,

$$\begin{aligned} g_{G,\eta}(\theta) &= -\ln\left(\sum_{x\in\{0,1\}^n} |\psi_x|^2 e^{-\eta f(x)}\right) \\ &= -\ln\left(\sum_{x\in\mathcal{F}} |\psi_x|^2 e^{-\eta f(x)} + \sum_{x\notin\mathcal{F}} |\psi_x|^2 e^{-\eta f(x)}\right) \\ &= -\ln\left(\left(\sum_{x\in\mathcal{F}} |\psi_x|^2\right) e^{-\eta f^*} + \sum_{x\notin\mathcal{F}} |\psi_x|^2 e^{-\eta f(x)}\right), \end{aligned}$$

where $f^*$ is the optimal value of $f$, reached on $\mathcal{F}$. By definition, $f(x) > f^*, \forall x \notin \mathcal{F}$, and because $\eta > 0$, we have $e^{-\eta f(x)} < e^{-\eta f^*}$. Moreover, we assume that $|\psi_{x_0}| \neq 0$, thus the second term of the sum in the logarithm is bounded above as follows: $\sum_{x\notin\mathcal{F}} |\psi_x|^2 e^{-\eta f(x)} < \left(\sum_{x\notin\mathcal{F}} |\psi_x|^2\right) e^{-\eta f^*}$, where $\sum_{x\notin\mathcal{F}} |\psi_x|^2 \geq |\psi_{x_0}|^2 > 0$. Thus, because $y \mapsto -\ln(y)$ is a decreasing function,

$$g_{G,\eta}(\theta) > -\ln\left(\left(\sum_{x\in\mathcal{F}} |\psi_x|^2\right) e^{-\eta f^*} + \left(\sum_{x\notin\mathcal{F}} |\psi_x|^2\right) e^{-\eta f^*}\right) = \eta f^*.$$

This contradicts the previous statement that $\theta \in \mathcal{G}$. Indeed, we can easily verify that the minimum of $g_{G,\eta}$ is $g^*_{G,\eta} = \eta f^*$. □

One might suggest other guiding functions, such as the minimum function

$$g_{\min}(\theta) = \min_{x\,:\,p_\theta(x)>0} f(x)\,. \tag{3.9}$$

However, $g_{\min}$ does not verify (3.3), and is not even continuous, hardening its optimization and excluding its choice for the guiding function.

**Example 8.** *Let us illustrate that $g_{min}$ is not a guiding function. For that, we consider the circuit of Figure 3.1 and the following function $f : \{0,1\}^3 \mapsto \mathbb{R}$ to minimize,*

$$\begin{cases} f(0,0,0) & = 1 \\ f(x) & = 0\,, \quad \forall x \neq (0,0,0) \end{cases}$$

*where $f^* = 0$ is the optimal value. Function $g_{min}$ reaches its optimal value $g^*_{min} = 0$ on the set of its optimizers*

$$\mathcal{G} = \mathbb{R}^3 \setminus \{(2k\pi, 2k\pi, 2k\pi) : k \in \mathbb{Z}\}\,.$$

*However,*

$$\forall \theta \in \mathcal{G} \setminus \{((2k+1)\pi, (2k+1)\pi, (2k+1)\pi) : k \in \mathbb{Z}\}, U(\theta)\,|0_n\rangle \notin \mathcal{F}_{quant}\,,$$

*because there is a non-zero probability of sampling $(0,0,0)$. Thus, $g_{min}$ violates (3.3). Moreover, $g_{min}$ is not continuous. Indeed, $g_{min}(0,0,0) = 1$, whereas $\forall \epsilon > 0\,, g_{min}(\epsilon,0,0) = 0$.*

The above observation motivated Barkoutsos et al. (2020) to suggest another function, the CVaR (Conditional Value-at-Risk) function. The CVaR function is the average on the lower $\alpha$-tail of values of $f$ encountered, where $\alpha \in ]0,1]$ is a parameter to be set. Let $(x_1, \ldots, x_{2^n})$ be the $n$-bitstrings sorted in non decreasing order, namely $f(x_i) \leq f(x_{i+1})$ for any $i \in [2^n - 1]$. Let $N_\alpha$ be the index that delimits the $\alpha$-tail elements of the distribution, specifically,

$$N_\alpha = \min \left\{ N \geq 1 : \sum_{i=1}^{N} p_\theta(x_i) \geq \alpha \right\} .$$

Then, the CVaR function is

$$g_{C,\alpha}(\theta) = \frac{1}{\sum_{i=1}^{N_\alpha} p_\theta(x_i)} \sum_{i=1}^{N_\alpha} p_\theta(x_i) f(x_i) . \tag{3.10}$$

The special case $\alpha = 1$ implies $g_\alpha = g_{\text{mean}}$, whereas when $\alpha$ approaches zero, we find $g_{\min}$.

The CVaR function is an alternative to the non-smooth minimum function. While CVaR does not verify (3.3) either, it keeps continuity and still focuses on the best solutions that appear on the probability distribution.

**Example 9.** *Let us illustrate the violation of* (3.3) *by the* CVaR *function, for any* $\alpha \in ]0,1[$. *For that, we consider function* $f$ *of Example 8 with the same quantum circuit of Figure 3.1. Let* $\alpha \in ]0,1[$. *Let us find* $\theta \in \mathcal{G}$ *such that* $U(\theta)\,|0_n\rangle \notin \mathcal{F}_{quant}$. *In other words, we search* $\theta \in \mathbb{R}^3$ *such that*

$$g_{C,\alpha}(\theta) = 0 \;\; \textit{and} \;\; p_\theta(0,0,0) > 0 .$$

*Let us look at* $\theta = (\pi - \epsilon, 0, 0)$, *where* $\epsilon \in ]0,\pi[$. *According to* (3.7), *we have*

$$p_\theta(0,0,0) = \cos^2\left(\frac{\pi - \epsilon}{2}\right) > 0 .$$

*It remains to choose* $\epsilon$ *to ensure* $g_{C,\alpha}(\theta) = 0$. *Namely, we want* $\epsilon$ *such that*

$$\sum_{x \neq (0,0,0)} p_\theta(x) \geq \alpha ,$$

*meaning*

$$1 - \cos^2\left(\frac{\pi - \epsilon}{2}\right) \geq \alpha .$$

*This holds for any* $\epsilon \leq \pi - 2\arccos\left(\sqrt{1-\alpha}\right)$.

Even if CVaR does not verify (3.3), it can be seen as a *pseudo*-guiding function, defined just below. Notice that in practice, using the CVaR function seems appropriate because it accepts a probability of measuring optimal solutions after optimizing to be lower than one, such as expressed in (3.2).

**Definition 3.1.6** (Pseudo-guiding function)**.** *Let* $g : \mathbb{R}^d \to \mathbb{R}$ *be a function and* $\mathcal{G}$ *be its set of minimizers. We call* $g$ *a pseudo-guiding function for* $f$ *with respect to* $U$ *if* $g$ *is continuous and if there exists* $\alpha \in ]0,1[$ *such that optima of* $g$ *can lead to non-optimal solutions of* $f$ *with a probability strictly lower than* $1 - \alpha$. *Specifically, let* $\theta \in \mathcal{G}$. *Thus, either* $U(\theta)\,|0_n\rangle \in \mathcal{F}_{quant}$ *or* $\sum_{x \notin \mathcal{F}} |\langle x|\, U(\theta)\,|0_n\rangle\,|^2 < 1 - \alpha$.

We show next that $g_{C,\alpha}$ is indeed a pseudo-guiding function according to Definition 3.1.6..

**Proposition 3.1.7.** *Let $\alpha \in ]0,1[$. Function $g_{C,\alpha}$ is a pseudo-guiding function.*

*Proof.* For the same reason as for $g_{\text{mean}}$, $g_{C,\alpha}$ is continuous.

Let $\alpha \in ]0,1[\,, \theta \in \mathcal{G}$ and let us consider the quantum state $|\psi(\theta)\rangle = U(\theta)\,|0_n\rangle$. Its decomposition in the canonical basis is $|\psi(\theta)\rangle = \sum_{x\in\{0,1\}^n} \psi_x\,|x\rangle$.

Let us prove that either $|\psi(\theta)\rangle \in \mathcal{F}_{\text{quant}}$ or that $\sum_{x\notin\mathcal{F}} |\psi_x|^2 = \sum_{x=N_{\mathcal{F}}+1}^{2^n} |\psi_{x_i}|^2 < 1-\alpha$, where $N_{\mathcal{F}}$ is the index that delimits $\mathcal{F}$, specifically,

$$\mathcal{F} = \{x_i, i \in [N_{\mathcal{F}}]\}\,.$$

There are two cases for $N_\alpha$ (we recall that $N_\alpha$ depends on $\theta$):

- If $N_\alpha \leq N_{\mathcal{F}}$, thus

$$\begin{aligned} g_{C,\alpha}(\theta) &= \frac{1}{\sum_{i\in[N_\alpha]\subseteq[N_{\mathcal{F}}]} |\psi_{x_i}|^2} \sum_{i\in[N_\alpha]\subseteq[N_{\mathcal{F}}]} |\psi_{x_i}|^2 f(x_i) \\ &= f^*\,, \end{aligned}$$

  where $f^*$ is the optimal value of $f$, reached on $\mathcal{F}$. This contradicts (3.3) but the probability of sampling non-optimal solutions when measuring $|\psi(\theta)\rangle$ is strictly lower than $1-\alpha$. Indeed, by definition of $N_\alpha$,

$$\sum_{i=1}^{N_{\mathcal{F}}} |\psi_{x_i}|^2 \geq \sum_{i=1}^{N_\alpha} |\psi_{x_i}|^2 \geq \alpha\,.$$

- Otherwise, $N_\alpha > N_{\mathcal{F}}$. Let us prove by contradiction that $|\psi(\theta)\rangle \in \mathcal{F}_{\text{quant}}$. Assume that $|\psi(\theta)\rangle \notin \mathcal{F}_{\text{quant}}$. Thus, there exists $k > N_{\mathcal{F}}$ such that $|\psi_k| \neq 0$. We can show that $g_{C,\alpha}(\theta) > f^*$ using essentially the same proof as that of Proposition 3.1.4 for $x_0 = \min\{k > N_{\mathcal{F}} : |\psi_k| \neq 0\}$. This contradicts the statement that $\theta \in \mathcal{G}$, because the minimum of $g_{C,\alpha}$ is $g^*_{C,\alpha} = f^*$.

□

#### 3.1.3.2 Classical optimizer

The role of the classical optimizer is to minimize the guiding function. The function $g$ is continuous and is usually differentiable but not convex. Any unconstrained optimization algorithm can be used to minimize $g$, such as local search, gradient descent method, or any black-box optimization algorithm.

To speak in terms of stochastic optimization, the classical optimizer aims at solving the stochastic programming model under endogenous uncertainty

$$\min_\theta \{g(\theta) = \mathbb{E}[G(\theta, \xi_\theta)]\}\,, \tag{3.11}$$

where the definition of $G$ depends on the choice of a specific guiding function, and $\xi_\theta$ is an endogenous vector that depends on $\theta$. Specifically, $\xi_\theta$ is a discrete random variable, with the set

of possible outcomes $\{0,1\}^n$ and the following distribution probability:

$$\mathbb{P}(\xi_\theta = x) = p_\theta(x), \quad \forall x \in \{0,1\}^n .$$

Thus, this problem falls into the class of stochastic dependent-decision probabilities problems (Hellemo et al., 2018). In practice, the classical optimizer approximates the function by a Monte Carlo estimation as follows:

$$\hat{g}_N(\theta) = \frac{1}{N} \sum_{j=1}^{N} G(\theta, \xi_\theta^j) ,$$

where $\{\xi_\theta^j\}_{j\in[N]}$ is a sample of size $N$ from the distribution of $\xi_\theta$. Notice that for a given $\theta$, the quantity $\hat{g}_N(\theta)$ itself is a random variable since its value depends on the sample that has been generated, which is random. In contrast, the value of $g(\theta)$ is deterministic. In practice, the classical optimizer iterates the loop that consists of, given a sampling distribution of size $N$ of the quantum state $U(\theta)\,|0_n\rangle$ (Remark 2.2.14), outputs a $\theta'$. This $\theta'$ is then transmitted to the quantum part. Eventually, the aim is to output $\theta^*$ such that $U(\theta)\,|0_n\rangle \subseteq \mathcal{F}_{\text{quant}}$. In each iteration, the value of $\hat{g}_N(\theta)$ is computed. Notice that according to the Law of Large Numbers (Shapiro, 2003), $\hat{g}_N(\theta)$ converges with probability one to $g(\theta)$ as $N \to \infty$.

For instance, for the case of $g = g_{\text{mean}}$, we have

$$G(\theta, \xi_\theta) = f(\xi_\theta) .$$

Hence, for each $\xi_\theta^j \in \{0,1\}^n$ sampled, either we compute classically $f(\xi_\theta^j)$ and store it if not already computed, or we get its value. Thus, we can compute $\hat{g}_N(\theta)$. Notice that for the CVaR function, we compute the empirical mean only on the best $\lceil \alpha N \rceil$ values $f(\xi_\theta)$ found by sampling.

The solution returned by the VQA is the minimum value of $f$ and the associated minimizer $x$ encountered while the algorithm runs.

## 3.2 Quantum Approximate Optimization Algorithm

We assume throughout this section that the function $f$ to be minimized is polynomial. First, we reformulate problem (3.1) to a more suitable form for quantum optimization. This reformulation is motivated by the quantum adiabatic evolution (Farhi et al., 2000) that, for a given Hermitian matrix[1], approximates the eigenvector with the lowest eigenvalue under certain conditions. For that, we interpret the objective function of problem (3.1) as an Hermitian matrix $H_f$ such that each eigenvector $|u_x\rangle$ is matching a classical solution $x \in \{0,1\}^n$ with an eigenvalue equal to $f(x)$, specifically,

$$H_f\,|u_x\rangle = f(x)\,|u_x\rangle \ .$$

Thus, the solutions of problem (3.1) are the solutions corresponding to the lowest eigenvalues of $H_f$. The Quantum Approximate Optimization Algorithm (QAOA) of Farhi et al. (2014) presented in this section aims at finding the lowest eigenvalue of $H_f$.

### 3.2.1 Problem reformulation

The construction of $H_f$ is as follows. First, we transform the $\{0,1\}$-problem (3.1) into a $\{-1,1\}$-problem. For that, we apply the following linear transformation: for $x = (x_1, \ldots, x_n) \in \{0,1\}^n$,

[1] A complex square matrix is Hermitian if it is equal to its conjugate transpose.

we define $z = (z_1, \dots, z_n) \in \{-1, 1\}^n$ where

$$z_i = 1 - 2x_i \,, \forall i \in [n] \,. \tag{3.12}$$

This leads to the problem

$$\min_{z \in \{-1,1\}^n} f_\pm(z) \,,$$

where, for $z \in \{-1, 1\}^n$,

$$f_\pm(z) = \sum_{\alpha=(\alpha_1,\dots\alpha_n)\in\{0,1\}^n} h_\alpha \prod_{i=1}^{n} z_i^{\alpha_i} \,,$$

where $h_\alpha \in \mathbb{R}, \forall \alpha \in \{0, 1\}^n$. Notice that, without loss of generality, we set $\alpha \in \{0, 1\}^n$ rather than $\alpha \in \mathbb{N}^n$, because the each variable $z_i$ is in $\{-1, 1\}$. Thus, only the parity of $\alpha_i$ matters in the term $z_i^{\alpha_i}$. Second, we define $H_f$ as

$$H_f = \sum_{\alpha=(\alpha_1,\dots\alpha_n)\in\{0,1\}^n} h_\alpha \bigotimes_{i=1}^{n} Z_i^{\alpha_i} \,,$$

where $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, $Z^0 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $Z^1 = Z$. We note $Z_i$ the application of $Z$ to qubit $i$. Notice that $Z$ is equal to the universal gate $R_Z(\pi)$ modulo a global phase (see Equation (2.6)). This construction of $H_f$ leads to the following property.

**Proposition 3.2.1.** *The eigenvectors of $H_f$ are the canonical basis $|x\rangle \in \mathcal{CB}_n$ with eigenvalues that are the cost of the solutions $f(x)$, specifically,*

$$\forall |x\rangle \in \mathcal{CB}_n, \;\; H_f |x\rangle = f(x) |x\rangle \;. \tag{3.13}$$

*Proof.* First, the eigenvectors of $H_f$ are the canonical basis states. Indeed, each term of the sum that constitutes $H_f$ is a tensor product of $n$ matrices $I$ or $Z$, both diagonal. Thus, $H_f$ is a $2^n$ diagonal matrix. Second, let us find the eigenvalues associated with the eigenvectors. Let $|x\rangle = |x_1 \dots x_n\rangle$ be in $\mathcal{CB}_n$. Let $z = (z_1, \dots, z_n)$ be the result of transformation (3.12). Thus, we can easily show that $Z |x_i\rangle = z_i |x_i\rangle \;, \forall i \in [n]$ and

$$\begin{aligned}
H_f |x\rangle &= \sum_{\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n} h_\alpha \bigotimes_{i=1}^{n} Z_i^{\alpha_i} |x_i\rangle \\
&= \sum_{\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n} h_\alpha \bigotimes_{i=1}^{n} z_i^{\alpha_i} |x_i\rangle \\
&= \left( \sum_{\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n} h_\alpha \prod_{i=1}^{n} z_i^{\alpha_i} \right) |x\rangle \\
&= f_\pm(z) |x\rangle \\
&= f(x) |x\rangle \;.
\end{aligned}$$

□

**Example 10.** *We illustrate this transformation on a small example for $n = 2$. Let us consider the problem*

$$\min_{x\in\{0,1\}^2} f(x) = x_1 + 2x_2 - 3x_1x_2\,.$$

*Using* (3.12)*, the equivalent $\{-1,1\}$-problem is*

$$\min_{z\in\{-1,1\}^2} f_{\pm}(z) = \frac{1}{4}z_1 - \frac{1}{4}z_2 - \frac{3}{4}z_1z_2 + \frac{3}{4}\,.$$

*Thus, the Hermitian matrix associated with the problem is*

$$H_f = \frac{1}{4}Z\otimes I - \frac{1}{4}I\otimes Z - \frac{3}{4}Z\otimes Z + \frac{3}{4}I\otimes I\,.$$

*To illustrate* (3.13)*, we compute the eigenvalue of the canonical basis state $|10\rangle$.*

$$\begin{aligned} H_f|10\rangle &= \frac{1}{4}(Z\otimes I)|10\rangle - \frac{1}{4}(I\otimes Z)|10\rangle - \frac{3}{4}(Z\otimes Z)|10\rangle + \frac{3}{4}(I\otimes I)|10\rangle \\ &= -\frac{1}{4}|10\rangle - \frac{1}{4}|10\rangle + \frac{3}{4}|10\rangle + \frac{3}{4}|10\rangle \\ &= |10\rangle \\ &= f(1,0)|10\rangle\ , \end{aligned}$$

*because $f(1,0) = 1$.*

Notice that most of the problems solved with QAOA in the literature are QUBO (Quadratic Unconstrained Binary Optimization) problems. Thus, $H_f$ has the specific form

$$H_f = \sum_i h_{ii}Z_i + \sum_{i<j} h_{ij}Z_i\otimes Z_j\,,$$

where $h_{ij}\in\mathbb{R}$, $\forall i\le j$. It is justified by the fact that solving QUBO problems to optimality is *already* NP-hard, and the quantum gates of the circuit are easier to implement on hardware in that case.

### 3.2.2 Quantum part

QAOA is a Variational Quantum Algorithm where the quantum part derives from the Hamiltonian $H_f$. This quantum part consists of a quantum circuit with $2p$ parameters $(\boldsymbol{\gamma},\boldsymbol{\beta}) = (\gamma_1,\dots,\gamma_p,\beta_1,\dots,\beta_p)\in\mathbb{R}^{2p}$, where $p$ is called *depth*. The quantum circuit $U(\boldsymbol{\gamma},\boldsymbol{\beta})$ is the sequence of $p$ layers of two blocks, initially applied to the uniform superposition $|+\rangle^{\otimes n}$ (see Example 5). To describe these blocks, we introduce the definition of a unitary operator associated with a Hamiltonian.

**Definition 3.2.2** (Unitary operator associated with Hermitian matrix)**.** *Given a Hermitian matrix $A$, we define its associated quantum gate* $\mathrm{Exp}(A,t)$ *parametrized by the parameter $t\in\mathbb{R}$ as*

$$\begin{aligned} \mathrm{Exp}(A,t) &= e^{-iAt} \\ &= \sum_{k=0}^{\infty}\frac{1}{k!}(-i)^k t^k A^k\,. \end{aligned}$$

*Because $A$ is Hermitian,* $\mathrm{Exp}(A,t)$ *is a unitary matrix.*

The first block is of the form $\mathrm{Exp}(H_f,\gamma)$, for $\gamma\in\mathbb{R}$, which is the unitary operator associated with the Hamiltonian $H_f$. The second block is of the form $\mathrm{Exp}(H_B,\beta)$, for $\beta\in\mathbb{R}$, which is the unitary operator associated with the Hamiltonian $H_B=\sum_{i=1}^{n}R_{X,i}(\pi)$ (see Equation 2.4 for the definition of rotation gate $R_{X,i}$). Thus, the quantum circuit is

$$U(\boldsymbol{\gamma},\boldsymbol{\beta})=\mathrm{Exp}(H_B,\beta_p)\mathrm{Exp}(H_f,\gamma_p)\dots\mathrm{Exp}(H_B,\beta_1)\mathrm{Exp}(H_f,\gamma_1)H^{\otimes n}\,, \tag{3.14}$$

where the first gates applied in this circuit are $H^{\otimes n}$ to then apply the $p$ layers to state $|+\rangle^{\otimes n}$. The three propositions that follow express the quantum circuit of QAOA with the set of universal gates (see Theorem 2.2.10). We give first the general decomposition of QAOA's quantum circuit defined in (3.14).

**Proposition 3.2.3.** *The first block* $\mathrm{Exp}(H_f,\gamma)$ *parametrized by* $\gamma\in\mathbb{R}$ *is*

$$\mathrm{Exp}(H_f,\gamma)=\prod_{\alpha\in\{0,1\}^n}\mathrm{Exp}(\bigotimes_{i=1}^{n}Z_i^{\alpha_i},h_\alpha\gamma)\,.$$

*The second block* $\mathrm{Exp}(H_B,\beta)$ *parametrized by* $\beta\in\mathbb{R}$ *is*

$$\mathrm{Exp}(H_B,\beta)=\bigotimes_{i=1}^{n}R_{X,i}(2\beta)\,.$$

*Proof.* Let $\gamma\in\mathbb{R}$ and let us consider the first block $\mathrm{Exp}(H_f,\gamma)$. By Definition 3.2.2, and because each pair of matrices of the family $\{\bigotimes_{i=1}^{n}Z_i^{\alpha_i}:\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n\}$ commutes two by two,

$$\begin{aligned}\mathrm{Exp}(H_f,\gamma)&=e^{-i(\sum_{\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n}h_\alpha\bigotimes_{i=1}^{n}Z_i^{\alpha_i})\gamma} &(3.15)\\ &=\prod_{\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n}e^{-ih_\alpha\bigotimes_{i=1}^{n}Z_i^{\alpha_i}\gamma} &(3.16)\\ &=\prod_{\alpha=(\alpha_1,\dots,\alpha_n)\in\{0,1\}^n}\mathrm{Exp}(\bigotimes_{i=1}^{n}Z_i^{\alpha_i},h_\alpha\gamma)\,. &(3.17)\end{aligned}$$

Let $\beta\in\mathbb{R}$ and let us consider the second block $\mathrm{Exp}(H_B,\beta)$. For more readability, we write $X_i=R_{X,i}(\pi)$ the application of matrix $X=\begin{pmatrix}0&1\\1&0\end{pmatrix}$ on qubit $i$. With the same development as above, and because each pair of matrices of the family $\{X_i:i\in[n]\}$ commutes two by two,

$$\mathrm{Exp}(H_B,\beta)=\prod_{i=1}^{n}\mathrm{Exp}(X_i,\beta)\,.$$

Let $i\in[n]$. Thus,

$$\mathrm{Exp}(X_i,\beta)=\sum_{k=0}^{\infty}\frac{1}{k!}(-i)^k\beta^kX_i^k \tag{3.18}$$

$$= \sum_{k=0}^{\infty} \frac{1}{(2k)!}(-i)^{2k}\beta^{2k}X_i^{2k} + \sum_{k=0}^{\infty} \frac{1}{(2k+1)!}(-i)^{(2k+1)}\beta^{(2k+1)}X_i^{(2k+1)} \tag{3.19}$$

$$= \sum_{k=0}^{\infty} \frac{1}{(2k)!}(-i)^{2k}\beta^{2k}I + \sum_{k=0}^{\infty} \frac{1}{(2k+1)!}(-i)^{(2k+1)}\beta^{(2k+1)}X_i \tag{3.20}$$

$$= \sum_{k=0}^{\infty} \frac{1}{(2k)!}(-1)^{k}\beta^{2k}I - i\sum_{k=0}^{\infty} \frac{1}{(2k+1)!}(-1)^{k}\beta^{(2k+1)}X_i \tag{3.21}$$

$$= \cos(t)I - i\sin(t)X_i \tag{3.22}$$

$$= R_{X,i}(2\beta)\,, \tag{3.23}$$

where Line (3.20) exploits the fact that $X^2 = I$, implying $X^{2k} = I$ and $X^{2k+1} = X$. Moreover, Line (3.21) applies the definition of the complex number $i$, and one can recognize the power series of cosinus and sinus functions. □

The particular case of QUBO is mainly considered in the literature. Thus, we propose next a decomposition of QAOA's quantum circuit for this specific case.

**Proposition 3.2.4.** *For the case of QUBO, the expression of* $\mathrm{Exp}(H_f, \gamma)$ *simplifies in*

$$\mathrm{Exp}(H_f, \gamma) = \left(\bigotimes_{i=1}^{n} R_{Z,i}(2h_{ii}\gamma)\right) \prod_{i<j} CX_{i,j}R_{Z,j}(2h_{i,j}\gamma)CX_{i,j}\,,$$

*and is rather easily implemented with universal quantum gates.*

*Proof.* Let $\gamma \in \mathbb{R}$. The application of (3.15)–(3.17) to the case of QUBO gives

$$\begin{aligned}
\mathrm{Exp}(H_f, \gamma) &= e^{-i(\sum_{i=1}^{n} h_{ii}Z_i + \sum_{i<j} h_{ij}Z_i\otimes Z_j)\gamma} \\
&= \prod_{i=1}^{n} e^{-iZ_i h_{ii}\gamma} \prod_{i<j} e^{-iZ_i\otimes Z_j h_{ij}\gamma} \\
&= \prod_{i=1}^{n} \mathrm{Exp}(Z_i, h_{ii}\gamma) \prod_{i<j} \mathrm{Exp}(Z_i \otimes Z_j, h_{ij}\gamma)\,.
\end{aligned}$$

Then, let us prove that, for $i \in [n]$ and $t \in \mathbb{R}$, $\mathrm{Exp}(Z_i, t) = R_{Z,i}(2t)$. The same development as above, Lines (3.20)–(3.22), replacing $X$ by $Z$ that have the same property $Z^2 = I$, gives

$$\mathrm{Exp}(Z_i, t) = \cos(t)I - i\sin(t)Z_i \tag{3.24}$$

$$= R_{Z,i}(2t)\,. \tag{3.25}$$

Eventually, Line (3.24) is the application of the gate $\begin{pmatrix} e^{-it} & 0 \\ 0 & e^{it} \end{pmatrix} = R_Z(2t)$ on qubit $i$, and the identity on the others.

It remains to prove that, for $i < j \in [n]$ and $t \in \mathbb{R}$, $\mathrm{Exp}(Z_i \otimes Z_j, t) = CX_{i,j}R_{Z,j}(2t)CX_{i,j}$. Following the same developments as above, we have

$$\mathrm{Exp}(Z_i \otimes Z_j, t) = \cos(t)I - i\sin(t)Z_i \otimes Z_j\,. \tag{3.26}$$

We consider the two-qubit system that corresponds to the qubit $i$ as the first qubit and the qubit

$j$ as the second qubit (the others are unchanged by the transformation). Thus, it remains to prove the equality of the two circuits depicted on Figure 3.2.

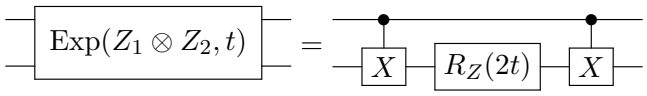


Figure 3.2: Decomposition of $\mathrm{Exp}(Z_1 \otimes Z_2, t)$ into universal gates.

On the one hand, (3.26) is the application of the gate $\begin{pmatrix} R_Z(2t) & 0 \\ 0 & R_Z(-2t) \end{pmatrix}$ to this system. Indeed,

$$\begin{aligned} \cos(t)I - i\sin(t)Z_1 \otimes Z_2 &= \cos(t)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - i\sin(t)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} e^{-it} & 0 & 0 & 0 \\ 0 & e^{it} & 0 & 0 \\ 0 & 0 & e^{it} & 0 \\ 0 & 0 & 0 & e^{-it} \end{pmatrix} \\ &= \begin{pmatrix} R_Z(2t) & 0 \\ 0 & R_Z(-2t) \end{pmatrix}. \end{aligned}$$

On the other hand, the composition of gates $CX_{1,2}R_{Z,2}(2t)CX_{1,2}$ on this system amounts to

$$\begin{aligned} CX_{1,2}R_{Z,2}(2t)CX_{1,2} &= \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix}\begin{pmatrix} R_Z(2t) & 0 \\ 0 & R_Z(2t) \end{pmatrix}\begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} \\ &= \begin{pmatrix} R_Z(2t) & 0 \\ 0 & XR_Z(2t)X \end{pmatrix} \\ &= \begin{pmatrix} R_Z(2t) & 0 \\ 0 & R_Z(-2t) \end{pmatrix}. \end{aligned}$$

Thus, the proof results from replacing $t$ by appropriate values $h_{ii}\gamma$ for $i \in [n]$ and $h_{ij}\gamma$ for $i < j$. □

The decomposition in universal quantum gates of the term $\mathrm{Exp}(Z_i \otimes Z_j, t)$ for the specific case of QUBO (see Proposition 3.2.4) is mainly used in the literature. We propose in Proposition 3.2.6 a generalization of such a decomposition for the term $\mathrm{Exp}(\bigotimes_{i=1}^{n} Z^{\alpha_i}, t)$, where the number of Z gates in effect can be bigger than two, namely, $|\{\alpha_i,\, i \in [n] : \alpha_i = 1\}| \geq 2$. This proposition enables us to overtake the QUBO problems, namely, to deal with a polynomial function $f$ with a degree strictly larger than two. We introduce next a technical result that will be necessary to derive the subsequent proposition.

**Lemma 3.2.5.** $\forall n \in \mathbb{N}^*$,

$$(I^{\otimes n-1} \otimes X)e^{-itZ^{\otimes n}}(I^{\otimes n-1} \otimes X) = e^{itZ^{\otimes n}}.$$

*Proof.* Let $P_n$ be the statement

$$(I^{\otimes n-1} \otimes X)e^{-itZ^{\otimes n}}(I^{\otimes n-1} \otimes X) = e^{itZ^{\otimes n}}\,.$$

Let us prove by induction that $P_n$ holds for all $n \in \mathbb{N}^*$.
**Base case:** Let us prove $P_1$. According to (3.25), $e^{-itZ} = R_Z(2t)$, thus,

$$\begin{aligned} Xe^{-itZ}X &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} e^{-it} & 0 \\ 0 & e^{it} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} e^{it} & 0 \\ 0 & e^{-it} \end{pmatrix} \\ &= e^{itZ}\,. \end{aligned}$$

**Induction step:** Let $n \geq 1$ be given and suppose $P_n$. Let us prove $P_{n+1}$. According to (3.27), we have $e^{-itZ^{\otimes n+1}} = \begin{pmatrix} e^{-itZ^{\otimes n}} & 0 \\ 0 & e^{itZ^{\otimes n}} \end{pmatrix}$. Thus,

$$\begin{aligned} &(I^{\otimes n} \otimes X)e^{-itZ^{\otimes n+1}}(I^{\otimes n} \otimes X) \\ &\quad = (I \otimes I^{\otimes n-1} \otimes X)e^{-itZ^{\otimes n+1}}(I \otimes I^{\otimes n-1} \otimes X) \\ &\quad = \begin{pmatrix} I^{\otimes n-1} \otimes X & 0 \\ 0 & I^{\otimes n-1} \otimes X \end{pmatrix} \begin{pmatrix} e^{-itZ^{\otimes n}} & 0 \\ 0 & e^{itZ^{\otimes n}} \end{pmatrix} \begin{pmatrix} I^{\otimes n-1} \otimes X & 0 \\ 0 & I^{\otimes n-1} \otimes X \end{pmatrix} \\ &\quad = \begin{pmatrix} (I^{\otimes n-1} \otimes X)e^{-itZ^{\otimes n}}(I^{\otimes n-1} \otimes X) & 0 \\ 0 & (I^{\otimes n-1} \otimes X)e^{itZ^{\otimes n}}(I^{\otimes n-1} \otimes X) \end{pmatrix}. \end{aligned}$$

By induction hypothesis $P_n$,

$$\begin{aligned} (I^{\otimes n} \otimes X)e^{-itZ^{\otimes n+1}}(I^{\otimes n} \otimes X) &= \begin{pmatrix} e^{itZ^{\otimes n}} & 0 \\ 0 & e^{-itZ^{\otimes n}} \end{pmatrix} \\ &= e^{itZ^{\otimes n+1}}\,. \end{aligned}$$

□

The proposition that follows enables a decomposition in universal quantum gates of any quantum circuit of QAOA.

**Proposition 3.2.6.** *Let us consider the subsystem composed of the $N$ qubits to which the $Z$ gate is applied. Specifically, $N = |\{\alpha_i\, i \in [n] : \alpha_i = 1\}|$, and we renumber the qubits in question in $[N]$. Thus, for $N \geq 2$, the term $\mathrm{Exp}(\bigotimes_{i=1}^{n} Z^{\alpha_i}, t)$ on this subsystem simplifies in*

$$\mathrm{Exp}(Z^{\otimes N}, t) = \prod_{j=0}^{N-2} CX_{1,N-j} R_{Z,N}(2t) \prod_{j=0}^{N-2} CX_{1,N-j}\,.$$

*We represent this decomposition on Figure 3.3.*

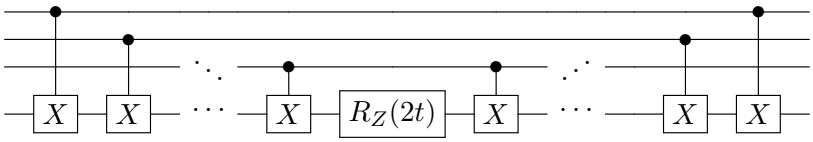


Figure 3.3: Decomposition of $\mathrm{Exp}(Z^{\otimes N}, t)$ on the $N$-qubit subsystem.

*Proof.* Let $P_n$ be the statement

$$\mathrm{Exp}(Z^{\otimes n}, t) = \prod_{j=0}^{n-2} CX_{1,n-j} R_{Z,n}(2t) \prod_{j=0}^{n-2} CX_{1,n-j}\,.$$

Let us prove by induction that $P_n$ holds for all $n \geq 2$.
**Base case:** Proposition 3.2.4 proves $P_2$.
**Induction step:** Let $n \geq 2$ be given and suppose $P_n$. Let us prove $P_{n+1}$.
On the one hand,

$$\mathrm{Exp}(Z^{\otimes n+1}, t) = e^{-itZ^{\otimes n+1}} = e^{-itZ \otimes Z^{\otimes n}}$$

where $Z \otimes Z^{\otimes n} = \begin{pmatrix} Z^{\otimes n} & 0 \\ 0 & -Z^{\otimes n} \end{pmatrix}$. This latter matrix is diagonal, thus,

$$\mathrm{Exp}(Z^{\otimes n+1}, t) = \begin{pmatrix} e^{-itZ^{\otimes n}} & 0 \\ 0 & e^{itZ^{\otimes n}} \end{pmatrix}. \tag{3.27}$$

On the other hand, we compute the term

$$\prod_{j=0}^{n-1} CX_{1,n+1-j} R_{Z,n+1}(2t) \prod_{j=0}^{n-1} CX_{1,n+1-j}$$
$$= CX_{1,n+1} \left( \prod_{j=1}^{n-2} CX_{1,n+1-j} R_{Z,n+1}(2t) \prod_{j=1}^{n-2} CX_{1,n+1-j} \right) CX_{1,n+1}$$

that is represented on Figure 3.4, where $e^{-itZ^{\otimes n}}$ applies on the qubits 2 to $n$ by induction hypothesis.

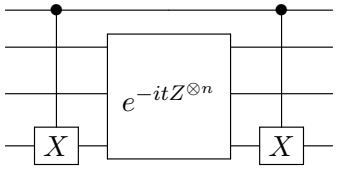


Figure 3.4: Circuit representation of $\prod_{j=0}^{n-1} CX_{1,n+1-j} R_{Z,n+1}(2t) \prod_{j=0}^{n-1} CX_{1,n+1-j}$.

Thus,

$$\begin{aligned}
&\prod_{j=0}^{n-1} CX_{1,n+1-j} R_{Z,n+1}(2t) \prod_{j=0}^{n-1} CX_{1,n+1-j} \\
&\qquad = CX_{1,n+1}(I \otimes e^{-itZ^{\otimes n}})CX_{1,n+1} \\
&\qquad = \begin{pmatrix} I^{\otimes n} & 0 \\ 0 & I^{\otimes n-1} \otimes X \end{pmatrix} \begin{pmatrix} e^{-itZ^{\otimes n}} & 0 \\ 0 & e^{-itZ^{\otimes n}} \end{pmatrix} \begin{pmatrix} I^{\otimes n} & 0 \\ 0 & I^{\otimes n-1} \otimes X \end{pmatrix} \\
&\qquad = \begin{pmatrix} e^{-itZ^{\otimes n}} & 0 \\ 0 & (I^{\otimes n-1} \otimes X) e^{-itZ^{\otimes n}} (I^{\otimes n-1} \otimes X) \end{pmatrix} \\
&\qquad = \begin{pmatrix} e^{-itZ^{\otimes n}} & 0 \\ 0 & e^{itZ^{\otimes n}} \end{pmatrix},
\end{aligned}$$

where the last line comes from the application of Lemma 3.2.5. Thus,

$$\mathrm{Exp}(Z^{\otimes n+1}, t) = \prod_{j=0}^{n-1} CX_{1,n+1-j} R_{Z,n+1}(2t) \prod_{j=0}^{n-1} CX_{1,n+1-j}\,,$$

proving $P_{n+1}$. □

**Example 11.** *We illustrate the construction of the quantum circuit with the problem of Example 10 where $n = 2$, for the case $p = 1$. Thus,*

$$U(\gamma, \beta) = \mathrm{Exp}(H_B, \beta)\mathrm{Exp}(H_f, \gamma) \left|+\right\rangle^{\otimes 2}, \quad \forall \gamma, \beta \in \mathbb{R}\,.$$

*The circuit is detailed in Figure 3.5.*

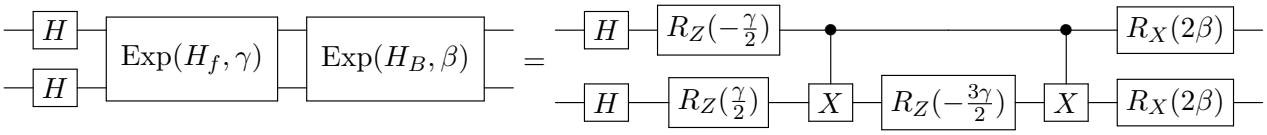


Figure 3.5: QAOA circuit of Example 10 for $p = 1$.

*Notice that we do not take into account the term $\frac{3}{4} I \otimes I$ of $H_f$ in this circuit. More generally, the term $h_{0\ldots0} I^{\otimes n}$ of $H_f$ never appears on the quantum circuit because it represents a constant term and does not influence the optimization.*

Notice that the choice of QAOA circuit does not ensure (3.4). For example, one can show that the probability of measuring 00 at the end of the circuit depicted in Figure 3.5 never reaches 1. Specifically,

$$p_{(\gamma,\beta)}(00) = |\left\langle 00\right| \mathrm{Exp}(H_B, \beta)\mathrm{Exp}(H_f, \gamma) \left|+\right\rangle^{\otimes 2}|^2 < \frac{1}{2}, \quad \forall \gamma, \beta \in \mathbb{R}\,.$$

However, QAOA satisfies another important property: it uses entangling gates. Each gate $\mathrm{Exp}(Z_i \otimes Z_j, t)$, for $t \in \mathbb{R} \setminus \{k\pi : k \in \mathbb{Z}\}$, entangles the qubits $i$ and $j$. Indeed, $\mathrm{Exp}(Z_i \otimes Z_j, t) = CX_{i,j} R_{Z,j}(2t) CX_{i,j}$, and unless $R_{Z,j}(2t) = I$, namely $t \in \{k\pi : k \in \mathbb{Z}\}$, the CNOT gate oper-

ates[2] and creates entanglement as mentioned in Subsubsection 2.2.4.3. The other gates, which are one-qubit gates, do not have this power. Entanglement is not necessary to (3.4). However, Remark 3.1.2 can justify the use of entanglement gate. Indeed, without it, it seems unlikely to achieve better results than pure classical optimization because the optimizer essentially solves a classical continuous relaxation.

In fact, the popularity of QAOA originates essentially from the fact that it mimics the adiabatic schedule (Wurtz and Love, 2022). The $n$-qubit system verifies the adiabatic condition when $p \to \infty$ and ensures that for a particular set of parameters, the quantum circuit gives the exact solution. However, quantum computers' quality today makes implementations on large instances impossible. But because the number of gates of the quantum circuit is $\mathcal{O}(pn^2)$, for small depth $d$, QAOA, and more generally VQAs, can already be implemented on current NISQ computers.

## 3.3 Literature review for QAOA

Many papers have recently addressed the empirical evaluation of QAOA, some also comparing it with specific implementations of VQAs. We present below a non-exhaustive list of these trends, we mention several theoretical limitations for specific cases known up to now and we end with the different leverages that are at stake to improve QAOA performances. This section does not provide a complete, up-to-date overview of QAOA performances, but rather aims at illustrating trends on combinatorial problems of interest to the operations research community.

### 3.3.1 Empirical and theoretical trends on QAOA

Let us begin with the numerical trends of QAOA performances. All the empirical experiments are presented on small instances because quantum computers' quality today makes implementations on large instances impossible, leading to difficult conclusions. Thus, many experiments are done on classical simulators of quantum computers. Most of the empirical results of QAOA apply to the MAX-CUT problem because it was initially the first application of QAOA Farhi et al. (2014).

**Definition 3.3.1** (MAX-CUT problem)**.** *Let $G = (V, E)$ be a undirected graph. A cut in $G$ is a subset $S \subseteq V$. We define its* cost *as the number of edges with one node in $S$ and one node in $V \setminus S$. The MAX-CUT problem aims at finding a cut with maximum* cost*. A version with weighted edges can also be defined.*

Notice that for the MAX-CUT problem, with the notations of Section 3.2, the objective function is

$$f(x) = -\sum_{(i,j)\in E} (x_i(1-x_j) + x_j(1-x_i)) \ ,$$

where $x_i$ is 1 if node $i$ is in the cut, 0 otherwise. The Hermitian matrix that corresponds to this problem is

$$H_f = -\frac{1}{2}\sum_{(i,j)\in E} (1 - Z_iZ_j) \, .$$

[2]For $t \in \{k\pi : k \in \mathbb{Z}\}$, $R_{Z,j}(2t) = I$. Thus, $CX_{i,j}R_{Z,j}(2t)CX_{i,j} = CX_{i,j}CX_{i,j} = I$ because CNOT is its own inverse.

The approximation ratio $r$ mainly quantifies the performance of QAOA as follows:

$$r = \frac{f^{\text{QAOA}}}{f^*},$$

where $f^{\text{QAOA}}$ is the value returned by QAOA, and $f^*$ is the optimal value. Papers often compare this ratio with the best-known guaranteed ratio of Goemans-Williamson algorithm Goemans and Williamson (1995), specifically, $r = 0.87856$. The seminal paper of QAOA Farhi et al. (2014) provides a lower bound of $r$ for the specific class of 3-regular graphs for $p = 1$ that is $r = 0.6924$. More precise analytical expressions of the lower bound of the ratio for $p = 1$ and for some other typical cases are given in Wang et al. (2018). In addition, the authors of Wurtz and Love (2021) provide lower bounds of the ratio for larger depth for uniform 3-regular graphs (under specific assumptions implying the absence of large cycles in the graph): $r \geq 0.7559$ for $p = 2$ and $r \geq 0.7924$ for $p = 3$.

Several empirical results on MAX-CUT spotlight patterns of optimal parameters and enable QAOA to exceed Goemans-Williamson bound for some specific instances. Some classes of MAX-CUT instances studied reveal patterns of optimal parameters. Thus, it seems to offer efficient heuristics for parameter selection and initialization. For example, the authors of Crooks (2018) look at the class of Erdös-Rényi graphs (random graphs where an edge appears between two nodes with probability 0.5) of size up to 17 nodes, where the classical optimizer is an automatic differentiation with stochastic gradient descent. The authors of Lotshaw et al. (2021) examine the exhaustive set of graphs with $n \leq 9$ nodes, with the gradient-based search BFGS (Broyden–Fletcher–Goldfarb–Shanno algorithm Fortran et al. (1992)) for the classical optimizer. Both exhibit instances that exceed the bound of Goemans-Williamson for small depth, $p \leq 8$ and $p \leq 3$, respectively. The sets of unweighted and weighted 3-regular graphs also lead to patterns in Zhou et al. (2020) for graphs of a maximum size of 22 nodes, also detected in the parameter space of the Job Shop Scheduling problem Kurowski et al. (2023). But even if the performance of QAOA sometimes exceeds the Goemans-Williamson bound for the low-depth circuits, it is believed that $p$ must grow with the instance size to have a chance to outperform the best classical algorithms. Indeed, for random large-girth d-regular graphs, QAOA with depth $p = 11$ presents better performances than any known classical algorithms, in the case where the optimal parameters are found and where $d$ goes to infinity Basso et al. (2021). Notice that the authors of the nominal paper of QAOA Farhi et al. (2014) tackle also the Sherrington-Kirkpatrick model, well-known in the spin glass theory. The Hamiltonian of this model represents the energy of $n$ spins with random coefficients, specifically,

$$H_f = \frac{1}{\sqrt{n}} \sum_{i<j} h_{ij} Z_i Z_j \,,$$

where the coefficients $h_{ij}$ are chosen independently from a distribution centered in 0 and with a variance equal to 1. In Farhi et al. (2022), it is shown that, for $p = 11$, QAOA outperforms asymptotically with $n$ the standard semidefinite programming algorithm and the spectral relaxation.

Some theoretical limits of QAOA are displayed, where the shape of the quantum state produced by the quantum circuit is at stake. The authors of Bravyi et al. (2020) point out that the symmetry and locality of this resulting variational state fundamentally limit the performances of QAOA. Indeed, they show that Goemans-Williamson outperforms QAOA for several instances of the MAX-CUT problem for any fixed depth $p$. Consequently, this paper suggests a non-local version of QAOA to overcome these limitations. The limits of the locality also appear when solving the

problem of the Maximum Independent Set (MIS). In Farhi et al. (2020), MIS instances are random graphs of $n$ vertices, with a fixed average degree $\bar{d}$. Thus, it proves that for depth $p \leq C \log(n)$, where $C$ is a constant depending on $\bar{d}$, QAOA cannot return an independent set better than 0.854 times the optimal for $\bar{d}$ large. Due to this locality issue, the author of Hastings (2019) compares QAOA with *local* classical algorithms which also have this locality notion: at each step, the value of a variable is updated depending on the values of its *neighbors*, i.e., the variables that share the same term within the objective function. Namely, after $t$ steps, the value of a variable depends on all information gathered in its *t-neighborhood*. Yet, these classical algorithms still outperform QAOA. A single step of these algorithms outperforms, resp. achieves the same performance as, a single step of QAOA ($p = 1$) for MAX-CUT instances, resp. MAX-3-LIN-2 instances. Notice that given binary variables and a set of linear equations modulo 2 with exactly three variables, the MAX-3-LIN-2 problem aims at finding a variable value assignment that maximizes the number of satisfied equations. Several other canonical combinatorial optimization problems have been tackled with QAOA in recent years. While these examples are limited to numerical tests on toy instances and do not include comparisons of performance with classical resolutions, we mention some of them to illustrate the growing interest in solving reference combinatorial problems with Variational Quantum Algorithms. For instance, in Radzihovsky et al. (2019), the authors reformulate the Traveling Salesman Problem (TSP) as an appropriate Hamiltonian matrix and test it on small instances of three and four cities. The authors of Tabi et al. (2020) reformulate and solve the Graph Coloring problem for a dozen of nodes. The authors of Kurowski et al. (2023) study the Job Shop Scheduling problem, find a suitable formulation, and implement it on an artificial instance with three machines and three jobs, each of them containing one or two operations. We end by citing a benchmark provided in Khumalo et al. (2022), that compares classical techniques, among them the Simulated Annealing, and quantum techniques including VQAs on NISQ quantum computers. It applies to the TSP and the Quadratic Assignment problem and shows that in terms of running time and quality solution, the classical methods significantly outperform the quantum ones. Notice that besides the latter paper, the mention of the computation time for VQAs in literature is rare. It could be explained by the fact that most of the experiments are tested on trivial instances and solved with quantum algorithms simulated on classical high-performance computers due to the noise on current quantum hardware De Palma et al. (2023).

### 3.3.2 Improvements and adaptations of QAOA

Despite the theoretical limitations displayed above, QAOA has leverages (guiding function, parametrized quantum circuit, classical optimizer, etc.) that are still of interest in the literature. We display some of these studies on different choices of leverages that empirically improve QAOA performances.

First, the guiding function is mainly the mean function (3.6) as in the seminal paper of QAOA. However, both the CVaR function (3.10) and the Gibbs function (3.8) give an alternative to the mean function and show empirical improvements. For the former, several optimization problems, such as MAX-CUT, Maximum Stable Set, MAX-3SAT, etc., are solved with QAOA in Barkoutsos et al. (2020) and show better results with faster convergence. For the latter, the authors of Li et al. (2020) display better results solving MAX-CUT with this guiding function. Notice that comparing these improvements is hard because they use different classical optimizers. A different method to guide the optimization, presented in Amaro et al. (2022) as a Filtering Variational Quantum Algorithm, substitutes the guiding function for *filtering functions*. A *filtering*

*function* is a function associated with a *filtering operator* that, given a Hamiltonian matrix and a quantum state, essentially modifies the latter by increasing the probability of eigenvectors with low eigenvalues and decreasing the probability of eigenvectors with high eigenvalues. Empirical results show improvements in the quality solution and the speed of convergence for the weighted Max-Cut problem.

Second, the choice of the quantum circuit is challenged in the literature. The underlying question is whether the quantum circuit of QAOA (see Subsection 3.2.2) is a good choice for finite depth $p$. Several papers suggest better circuits based on empirical results for small depth and small instances. For example, VQA with the circuit proposed in Barkoutsos et al. (2020) has better performances than QAOA, or with the *Bang-Bang* circuit described in Yang et al. (2017). The benefit of entanglement gates in the circuit is also discussed in Nannicini (2019) without giving a clear advantage. One can refer to the survey Blekos et al. (2024) for other variants of quantum circuits. Notice that the authors of Egger et al. (2021) suggest to warm-start QAOA with either a continuous relaxation or a randomized rounding. This consists of initializing the quantum circuit with a solution of a continuous relaxation (Quadratic Programming or Semi Definite Programming), respectively with a randomly rounded solution of a continuous relaxation, instead of the state $|+\rangle^{\otimes n}$. In both cases, QAOA performances for small $p$ are better with a warm-start. Other initialization techniques are proposed in the literature. Among them, the annealing-inspired method Sack and Serbyn (2021) shows that a random parameters initialization on random graphs for the MAX-CUT problem is outperformed by the use of a Trotterized Quantum Annealing initialization. Another proposition is to initialize the parameters by using Machine Learning models Alam et al. (2020). Specifically, looking for Machine Learning models that enable the prediction of parameters close to the optimal is relevant because there is an observed correlation between parameters of low-depth and high-depth circuits for some problems. For instance, this type of initialization empirically reduces the number of iterations of the classical optimizer of 44.9% on average for MAX-CUT problems.

The choice of the classical optimizer represents another leverage, where both gradient-based and gradient-free optimizers can be used. The most encountered gradient-based methods used in VQAs in the literature are the Gradient Descent Ruder (2016), the Broyden–Fletcher–Goldfarb-Shanno algorithm Fortran et al. (1992), and the stochastic optimization algorithm ADAM Kingma and Ba (2014). As gradient-free methods, we can find the constrained optimization with linear approximation algorithm COBYLA Powell (1994), the simplex-based algorithm Nelder-Mead Nelder and Mead (1965), and the simultaneous perturbation stochastic approximation algorithm SPSA Spall (1992). The authors of Nannicini (2019) advise the choice of a global optimizer rather than a local optimizer to avoid numerous local optima. In Soloviev et al. (2022), the authors choose the gradient-free evolutionary algorithm called Estimation of Distribution Algorithm as the classical optimizer. This algorithm generates new solutions from a probabilistic model that depends on the best solutions of the previous iterations. They empirically show that it improves the results compared to traditional optimizers as mentioned above. The expression of the quantum circuit can also produce barren plateaus, hardening the optimization McClean et al. (2018); Holmes et al. (2022). However, the authors of Mastropietro et al. (2023) indicate that re-starting the optimization when reaching barren plateaus, where the new parameters are chosen using a stochastic process, improves QAOA performances on MAX-CUT problems. Notice that the classical optimization loop is not always required. Indeed, the authors of Brandao et al. (2018) express that for some classes of QUBO problems, the (near) optimal parameters do not depend on the instance, allowing to train parameters before executing the quantum circuit only once. This is called the *concentration of parameters* and is numerically illustrated on random

3-regular graphs of 20 nodes for low-depth circuits regarding the MAX-CUT objective function. We do not list all the possible choices of VQA leverages. The surveys Cerezo et al. (2021); Blekos et al. (2024) propose other possibilities.

In parallel, several algorithms derived from QAOA are appearing in the literature. An adaptation of QAOA is the Recursive-QAOA Bravyi et al. (2022), also called RQAOA. It applies first several times QAOA to the problem in order to reduce its size, namely the number of variables, according to the correlation that appears between some variables. Then, it solves with classical brute force the resulting smaller problem. This algorithm seems to be competitive compared to QAOA for problems such as MAX-k-CUT. The authors of Zhu et al. (2022) propose another algorithm, the Adaptive-QAOA, that converges faster than QAOA on some instances of MAX-CUT. It consists of applying recursively QAOA, increasing step by step the depth of the quantum circuit.

Eventually, there is ongoing work on the formulation and implementation of the QAOA circuit to ease and lighten QAOA implementation. For instance, the authors of Nüßlein et al. (2022) present an algorithmic method to reduce the growth of the QUBO matrix size with the problem size for the $k$-SAT problem and the Hamiltonian Cycles problem. Besides that, the authors of Herrman et al. (2021) expose a *global variable substitution method* that, given an initial linear formulation of a 3-SAT problem, exploits the advantage of product representation in QAOA and, thus, minimizes the circuit depth[3]. The shallower depth, the more efficient the implementation of the quantum circuit for NISQ computers. A different approach is proposed in Nagarajan et al. (2021), that is, given a quantum circuit, minimizes the circuit depth by solving a Mixed-Integer Program, with an optimality guarantee on the quantum circuit produced (it can find up to 57% reduction of the number of gates). It applies to any gate-based quantum algorithm and thus, can be applied to QAOA and, more generally, to VQAs.

Notice that solving combinatorial optimization problems using QAOA involves first formulating them into unconstrained problems, and more precisely into QUBO for easier circuits. There are essentially two types of formulation. The first and most common one is to integrate constraints as suitable penalty terms into the objective function. For instance, in Lucas (2014) we find formulations of Karp's 21 NP-complete problems, and the authors of Oh et al. (2019) tackle the $k$-coloring graph problem with the same method. The tutorial Glover et al. (2022) addresses more general cases for this formulation problem methodology. The second type of formulation is to change the expression of the *mixing* Hamiltonian (referred to as $H_B$ in Section 3.2). Initially presented in Hadfield et al. (2019), the main idea is to make this Hamiltonian varying the quantum state only in the feasible search space. Some problems have been specifically studied with this method, such as the Traveling Salesman problem Ruan et al. (2020); Radzihovsky et al. (2019). Notice that there exists a third way to deal with constrained problems, which relies on the quantum Zeno dynamics. This consists of a generic method that essentially projects the quantum states into the feasible space Herman et al. (2023). This requires auxiliary qubits which are measured in the middle of VQA or QAOA circuit. This shows improvements of QAOA's performances, compared to the method with penalty terms, for the portfolio optimization problem numerically assessed on instances with less than ten assets.

[3]Notice that here we talk about the general depth, not $p$, which is the longest path in the circuit, namely the maximum number of gates executed on a qubit.

## 3.4 Constrained integer optimization

In Section 3.1, we showed that VQAs tackle Polynomial Unconstrained Binary Optimization (PUBO) problems. However, most real-world combinatorial problems are constrained. Some constraints are directly related to the definition of the problem, for instance, that a city is visited exactly once by the salesman in the TSP. Some others express real-world limits, such as the limited number of seats on a train or the finite size of a knapsack, involving numerical constants in the description of the problem. In both cases, we need to reformulate the problem as an unconstrained optimization problem to solve it with VQAs. To remove the constraints, we integrate them as penalty terms in the objective function. As mentioned in the previous section, this approach is commonly used in the literature, as presented by Glover et al. (2022) or by Lucas (2014) to reformulate several NP-hard problems into QUBO problems. Notice that other techniques to remove constraints have been proposed, such as modifying the circuit of VQAs to *express* the constraints (Hadfield et al., 2019). As far as we know, the interest of the operational research community in these reformulations (by penalizing unfeasible solutions in their costs) focuses only on QUBO problems because this model can be solved on the majority of current quantum architectures (gate-based quantum computers and photonic computers with variational algorithms, or the analog computer of Dwave with quantum annealing).

In this section, we aim to widen the range of problems we can address with VQAs. For that, we provide a general method to transform any problem with integer variables, a polynomial objective function, and polynomial constraints into a PUBO problem.

### 3.4.1 Integration of constraints

First, we present a generic way to remove constraints from a nominal constrained problem and ensure that the reformulation into the resulting unconstrained problem is valid. Specifically, let us consider the problem

$$\begin{aligned} &\min_{x\in\mathcal{X}} && f(x) && \text{(Nominal)}\\ &\text{subject to} && \text{the set of constraints } \{k\in\mathcal{K}\} \end{aligned}$$

For each constraint $k \in \mathcal{K}$, we define a penalty function $pen_k : \mathcal{X} \to \mathbb{R}^+$ that satisfies, for $x \in \mathcal{X}$,

$$pen_k(x) \begin{cases} = 0 & \text{if } x \text{ satisfies constraint } k \\ \geq 1 & \text{if } x \text{ violates constraint } k \end{cases}$$

We reformulate the constrained problem (Nominal) as follows.

**Definition 3.4.1.** *The unconstrained problem, for which we integrate the constraints of the nominal problem as penalty terms in the objective function, is*

$$\min_{x\in\mathcal{X}} f(x) + \sum_{k\in\mathcal{K}} \lambda_k pen_k(x)\,, \qquad \text{(Unconstrained)}$$

*where the $\lambda_k > 0$ are penalty coefficients.*

Thus, the reformulation of (Nominal) into (Unconstrained) not only requires finding the penalty functions $pen_k$, as we will discuss in Subsection 3.4.2, but also needs to set the values of the penalty coefficients $\lambda_k$. Next, we provide a general lower bound for them.

Let us note $f_{\min} := \min\{f(x) : x \in \mathcal{X}\}$ the minimum value of $f$, $f_{\max} := \max\{f(x) : x \in \mathcal{X}\}$ its maximum value, and $f^* := \min_{x \in \mathcal{X}}\{f(x) : x \text{ respects constraint } k, \forall k \in \mathcal{K}\}$ the optimal value of (Nominal).

**Proposition 3.4.2.** *If we set, for all $k \in \mathcal{K}$,*

$$\lambda_k \geq f_{max} - f_{min}\,,$$

*thus, we ensure that solving* (Nominal) *amounts to solving* (Unconstrained)*.*

*Proof.* The smallest value of an unfeasible solution of (Unconstrained) is always larger than the minimum value of $f$ plus the penalty cost of violating at least one constraint $k'$, namely, larger than

$$f_{\min} + \lambda_{k'}\,.$$

Thus, for $\lambda_k \geq f_{\max} - f_{\min}$ for any constraint $k$, the smallest value of an unfeasible solution of (Unconstrained) is larger than

$$f_{\min} + \lambda_{k'} \geq f_{\min} + f_{\max} - f_{\min} = f_{\max} > f^*\,.$$

Moreover, by definition of the penalty function, the values of the objective function of the two problems coincide on feasible solutions. Consequently, the optimal value of (Nominal), and its corresponding solution, is equal to the one of (Unconstrained). □

Notice that in practice, an upper bound of $f_{\max}$ provides a lower bound for each $\lambda_k$. In this case, and if $f_{\min} \geq 0$, any feasible solution of (Unconstrained) has a lower objective function value than any unfeasible solution of (Unconstrained). This is worth noting because VQAs are heuristics, so they do not always find the optimal solution but solutions close to the optimal (in terms of the loss function). Thus, setting $\lambda_k \geq f_{\max}$ ensures that these solutions are feasible.

In this subsection, we showed that reformulating a constrained problem into an unconstrained problem amounts to finding the penalty functions for each constraint. Next, we present a broad class of problems for which we provide the expression of the penalty functions.

### 3.4.2 Class of eligible problems

We present a class of problems (IP-poly) for which we provide next a method to reformulate them as PUBO problems. This class contains problems with integer variables, a polynomial objective function, and polynomial constraints.

**Definition 3.4.3.** *Let $n \in \mathbb{N}$ be the number of variables and let $m \in \mathbb{N}$ be the number of constraints. We call* (IP-poly) *the following class of problems.*

$$\begin{aligned} \min_{x} \quad & f(x_1, \ldots, x_n) && && \text{(IP-poly)} \\ \text{subject to} \quad & g_k(x_1, \ldots, x_n) \leq 0\,, && \forall k \in [m] \\ & x_i \in \mathbb{N}, && \forall i \in [n] \end{aligned}$$

*where the functions $f$ and $g_k$ for any $k \in [m]$, are polynomial. Specifically,*

$$\min_{x} \quad \sum_{\gamma=(\gamma_1,\ldots,\gamma_n)\in\Gamma} \alpha_\gamma x_1^{\gamma_1} \ldots x_n^{\gamma_n} \qquad \text{(IP-poly)}$$

$$\text{subject to} \quad \sum_{\gamma=(\gamma_1,\dots,\gamma_n)\in\Gamma_k} \beta_{k,\gamma} x_1^{\gamma_1}\dots x_n^{\gamma_n} \leq 0\,, \qquad \forall k\in[m] \qquad (\mathrm{C}_k)$$
$$x_i \in \mathbb{N}, \qquad \forall i\in[n]$$

*where $\Gamma \subseteq \mathbb{N}^n$ is a finite set and $\alpha_\gamma \in \mathbb{R}$ for $\gamma\in\Gamma$; and for all $k\in[m]$, $\Gamma_k \subseteq \mathbb{N}^n$ is a finite set and $\beta_{k,\gamma}\in\mathbb{Z}$ for $\gamma\in\Gamma_k$.*

### 3.4.3 Transformation into PUBO

The two steps to transform any problem of (IP-poly) into a PUBO problem are the following. First, we transform the integer variables into binary variables. Second, we integrate the constraints into the objective function as penalty terms. Let us specify each of these steps.

**Transformation of integer variables into binary variables** We replace each integer variable $x_i$, for $i\in[n]$, by its binary decomposition

$$x_i = \sum_{j=0}^{\lfloor \log_2(x)\rfloor} x_i^{(j)} 2^j\,.$$

Thus, this decomposition requires $\lfloor\log_2(x)\rfloor+1$ binary variables $x_i^{(j)}\in\{0,1\}$.

**Integration of the constraints into the objective function as penalty terms** Let us consider the constraint $(\mathrm{C}_k)$

$$\sum_{\gamma=(\gamma_1,\dots\gamma_n)\in\Gamma_k} \beta_{k,\gamma} x_1^{\gamma_1}\dots x_n^{\gamma_n} \leq 0\,. \qquad (\mathrm{C}_k)$$

This step aims to find a penalty function for the above-mentioned constraint. Notice that after the first step, all variables are binary, so the integer variables of the left-hand side of $(\mathrm{C}_k)$ would be replaced by their binary description, adding more terms to the sum. To ease the reading, we assume henceforth that all the $x_i$ are binary variables. It results the following upper bound

$$\left|\sum_{\gamma\in\Gamma_k} \beta_{k,\gamma} x_1^{\gamma_1}\dots x_n^{\gamma_n}\right| \leq \sum_{\gamma\in\Gamma_k} |\beta_{k,\gamma}| =: \mathrm{UB}_k\,.$$

Thus, we define the penalty function associated with Constraint $(\mathrm{C}_k)$ as follows.

**Proposition 3.4.4.** *For $k\in[m]$, the function*

$$pen_k(x_1,\dots,x_n) = \prod_{j=0}^{\mathrm{UB}_k}\left(\sum_{\gamma\in\Gamma_k}\beta_{k,\gamma}x_1^{\gamma_1}\dots x_n^{\gamma_n} + j\right)$$

*is a penalty function for Constraint $(\mathrm{C}_k)$.*

*Proof.* On the one hand, if $(x_1,\dots,x_n)$ satisfies the constraint, it means that $\sum_{\gamma\in\Gamma_k}\beta_{k,\gamma}x_1^{\gamma_1}\dots x_n^{\gamma_n}$ takes a value in $[\![-\mathrm{UB}_k, 0]\!]$ and then there exists $j\in[\![0,\mathrm{UB}_k]\!]$ that makes the product equal to 0, i.e. $pen_k(x_1,\dots,x_n)=0$. On the other hand, if $(x_1,\dots,x_n)$ violates the constraint, the term $\sum_{\gamma\in\Gamma_k}\beta_{k,\gamma}x_1^{\gamma_1}\dots x_n^{\gamma_n}$ is strictly positive. Precisely, because each $\beta_{k,\gamma}$ is in $\mathbb{Z}$, the term $\sum_{\gamma\in\Gamma_k}\beta_{k,\gamma}x_1^{\gamma_1}\dots x_n^{\gamma_n}$ cannot be small than 1, leading to $pen_k(x_1,\dots,x_n)\geq 1$. □

## 3.5 Conclusion

In this chapter, we studied the class of Variational Quantum Algorithms, and the Quantum Approximate Optimization Algorithm (QAOA) in particular, which are hybrid heuristics and represent the first numerical applications of quantum computing for solving combinatorial problems. Because these algorithms address unconstrained problems, we also proposed a generic method to reformulate constrained problems into unconstrained ones. In the next chapter, we will tackle a railway timetabling problem of SNCF with QAOA, illustrating both the reformulation and the resolution stakes.

# 4

# Application to a railway timetabling problem

SNCF has to handle numerous and hard decision-making and optimization challenges, which directly impact the company's performance in terms of production efficiency, operational performance, and service quality. Currently, SNCF uses operations research techniques to solve combinatorial optimization problems. While exact methods like linear programming ensure finding the optimal solution, the complexity of the problems and the size of the instances often require dividing the problems into sub-problems and/or using heuristics. However, these types of resolutions partially degrade the solutions. The resolution of *strategical* problems is particularly decisive, as they precede *pre-operational* and *real-time* problems. In this chapter, we focus on one of them: the railway timetabling problem, for which the model considered by SNCF is NP-hard. This choice has been guided not only by the current difficulty of solving it with classical methods but also by its scalability. Indeed, the time scale and geographical perimeter of the planning can be easily adjusted, making its study on small instances still relevant.

In this chapter, we describe the railway timetabling problem at hand and introduce two simplified models (SCP and Extended-BP), each varying in precision, to make its resolution with current gate-based quantum machines possible. For each model, we provide a reformulation in Quadratic Unconstrained Binary Optimization (QUBO) and Polynomial Unconstrained Binary Optimization (PUBO) problems. Eventually, we illustrate the performance of the Quantum Approximate Optimization Algorithm (QAOA) for these different reformulations, analyzing the influence of QAOA's parameters, on both random and small real instances.

The work on the second simplified model, (Extended-BP), has been a joint work with Marion Lavignac (intern at SNCF).

## 4.1 Railway timetabling problem

The railway timetabling problem is a crucial problem for railway companies. Indeed, a first version of the timetable is planned several years in advance, and is related to other planning problems, such as crew scheduling or rolling stock scheduling. The goal of timetabling is to ensure the satisfaction of customers, the minimization of delays thanks to robustness and resilience properties, the minimization of costs, and the validation of both operational and security constraints. The railway timetabling problem we are considering is the following: according to the customers' demand (estimated from past data) and the availability of the network and the rolling stock, the aim is to find the transportation plan maximizing the operating profit. The output is a timetable of trains, the associated rolling stock schedule, and a forecast of the passengers for each train and journey. The optimal solution is the best compromise between the revenues generated by the customers' journeys and the production costs (network, rolling stock, human resources, etc.).

The combinatorial complexity of this problem, which is NP-hard, prevents its efficient resolution

by current classical computers on large geographical perimeters. It is a key problem for SNCF because reducing the computation time, improving the solution's quality, and considering large perimeters would improve significantly both the customers' experience and the benefits for the company. The perimeters at stake for this problem are in the range of the size of a region for the following reasons. First, the timetabling is historically built by regional poles. Second, planning at the national level should bring significant improvements to the resulting timetabling, but, as mentioned earlier, this is not currently possible with classical methods.

Many kinds of timetabling problems are studied in the railway research community. Allocation of customers to available trains, distribution of resources to trains on the railway network, and so on lie at the core of the railway work. Such problems are often demonstrated as hard problems in complexity theory and have therefore raised the scientific community's interest. Today, timetabling problems are solved with different methods as listed by Chen et al. (2021): operational research-based techniques such as graph coloring solvers, meta-heuristics like tabu search or simulated annealing, and many others. Moreover, quantum algorithms have been recently investigated and applied to different railway or transportation problems such as seating arrangement (Gioda, 2021), rolling stock planning (Bickert et al., 2021) or traffic navigation (Yarkoni et al., 2020). The proposed resolutions are executed with the quantum annealing machine of DWave to deal with instances for which the encoding requires more than a hundred qubits, which is the average size of current gate-based quantum computers. As far as we know, these types of resolutions do not show any quantum advantage at this time.

In this section, we propose a resolution of our timetabling with the class of Variational Quantum Algorithms presented in Chapter 3 and more precisely with QAOA detailed in Section 3.2. This choice is driven by two aspects. First, we want to explore the empirical part of these metaheuristics we studied theoretically previously. Second, we believe that the gate-based model is worth investigating for the optimization field because this is a model executing most of the theoretical algorithms already designed, proving its interest for long-term considerations.

### 4.1.1 Nominal problem at SNCF

The railway timetabling problem for high-speed trains at SNCF is formulated as an Integer Linear Programming (ILP). For the sake of confidentially, we provide a basic description of it while explaining the important ideas to understand the various simplifications proposed below.

Let us first describe the main sets of an instance of our problem.

- S, the set of Train-paths. A train-path is a timed unitary portion of tracks. It defines the availability to run a carriage over a portion of tracks over a given time period.
- T, the set of available Trains. A train is described as a union of train-paths. A train is defined by its origin, destination, and the served stations, with departure and arrival times for each station. A train uses the same carriage throughout the journey.
- G, the set of Groups of customers. A group gathers customers that have the same preferences on journeys, namely, customers wanting to leave, respectively arrive, at the same station and at the same time.
- $R = \{(t, g) \in T \times G : \text{group } g \text{ accepts to take train } t\}$, the set of possible Customers' journeys. This set expresses the possibilities to satisfy the customer groups' demands.

Other sets are required, such as the set expressing the different incompatibilities between train-paths or the set of trains that can be coupled. Besides sets, many constants appear in the original

formulation, both in the objective function and in the constraints, such as the maximum capacity of a carriage, the toll cost of a train-path, or the average receipt for a journey.

Second, let us introduce the variables of this problem, which are binary or integers.

- $x \in \{0,1\}^{|T|}$, where $x_t = 1$ iff $t \in T$ is used in the timetable
- $u \in \{0,1\}^{|S|}$, where $u_s = 1$ iff the train-path $s \in S$ is used
- $z \in \mathbb{N}^{|R|}$, where $z_r$ is the number of customers for journey $r \in R$

The objective function leads to finding the timetable providing the best compromise between customers' demand satisfaction and production cost. Specifically,

$$\max_{x_t, u_s, z_r} \alpha(z_r) - \beta(x_t, u_s)\,,$$

where $\alpha(z_r)$ is a linear function representing the receipts generated by selling tickets and $\beta(x_t, u_s)$ is a linear function representing the total cost associated with the use of train-paths and carriages. An optimal solution to our problem is a feasible timetable that maximizes the above loss function. The feasibility of a timetable is defined by many linear constraints such as forbidding customers to take a train not used in the timetable, ensuring that the maximum capacity of a train is satisfied on each train-path, or setting a minimum daily frequency for a given journey. In practice, this railway timetabling problem requires about twenty sets and constants to define an instance of the nominal problem, while it requires four types of binary or integer variables, a linear function, and ten types of linear constraints, two of which are soft constraints. Today, SNCF solves it with the CPLEX solver and, for many instances, rarely ends in finding the optimal solution. For example, the optimal solution of the problem for the sector between Paris and Lyon is found within a second whereas the solution for timetabling of the inter-regional trains has a gap of 67% from the optimal solution after 10 minutes of running. No solution is found for larger instances such as the perimeter of the entire metropolitan French territory.

To solve this railway timetabling problem with quantum-classical metaheuristics on current quantum machines, or classical simulators of quantum machines, we need to simplify it by putting aside some assumptions. Indeed, the size of the problem would be too large even considering instances on small sectors and for a short period (several days). For example, if we consider the instance on the sector Paris-Lyon for on day only, which covers 6 stations, it amounts to 176 customer groups for around 25 000 customers, 87 trains, 157 train-paths leading to a nominal problem with 8 000 binary variables. Additionally, Variational Quantum Algorithms require unconstrained problems (QUBO or PUBO problems), integrating constraints in the objective function costs in terms of additional variables (for QUBO reformulation) and additional gates (for QUBO and PUBO reformulation) as explained in Chapter 3. As an example, the QUBO formulation of the above-mentioned instance requires roughly 12 000 additional qubits, ending up to 20 000 qubits for the total description of the instance. This size prohibits a resolution on current gate-based quantum hardware that does not exceed a hundred qubits, without mentioning the high connectivity that would be necessary. While keeping the essence of the initial problem of timetabling, our simplification allows formulating the problem as a Set Cover Problem, or as an extended version of a Bin Packing problem, as we detail next.

### 4.1.2 Simplification to Set Cover and Extended Bin Packing problems

In this subsection, we present two simplifications of the nominal railway problem, first in (SCP) which is simpler and second in (Extended-BP) which gets closer to the nominal problem, although

still simplified.

**Set Cover Problem simplification** Let us present the simplified version of the nominal problem that results in a Set Cover Problem (SCP). This simplification highly reduces the feasibility and the quality of the solution but maintains the core of the original problem. As mentioned earlier, this simplification is essentially motivated by two things. The first one is to reduce the number of qubits to describe an instance. The second one is to ease the transformation of the constrained problem into an unconstrained one, avoiding adding too many qubits for the description of the new problem, and too many gates for the implementation of the quantum circuit of VQAs. Moreover, the SCP is NP-hard as the nominal problem, which comforts the interest of this choice.

The simplification is done as follows. First, we approximate the production cost of a timetable by the number of trains it contains. Second, we impose that each customer groups' demand must be satisfied, namely that each group takes a train on its demand (i.e. a train that it accepts). This leads to the resulting SCP below.

Let $m \in \mathbb{N}$ be the number of customer groups and $n \in \mathbb{N}$ the number of available trains. We define

- $G := \{g_1, \dots, g_m\}$, a set of $m \in \mathbb{N}$ customer groups.
- $T := \{T_1, \dots, T_n\}$, the set of $n \in \mathbb{N}$ available trains, where each train $T_i \subseteq G$ is the set of groups that accept to take this train.

We consider that a customer group $g_k$, for $k \in [m]$, is satisfied if at least one of the trains matching its demand (the $T_i$ such that $g_k \in T_i$) is in the output timetable.

In this setting, a customer group is an element and a train is a subset of the SCP. The SCP aims at finding the minimum number of subsets covering all the elements. Thus, an optimal solution of SCP is a set of trains of minimum cardinality such that each customer group is satisfied. Mathematically, it is formulated as follows.

**Definition 4.1.1** (Set Cover Problem)**.** *We consider the Set Cover Problem*

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{n} x_i && && \text{(SCP)} \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ki} x_i \geq 1, && \forall k \in [m] && \text{(Satisf)} \\ & x_i \in \{0,1\}, && \forall i \in [n] \end{aligned}$$

*where the binary variables indicate which available train is taken in the timetable, namely,* $\forall i \in [n]$,

$$x_i = \begin{cases} 1 \text{ if train } T_i \text{ is used in the timetable} \\ 0 \text{ else} \end{cases}$$

*and the coefficients of the constraints express the customer demands, namely,* $\forall i \in [n]$, $\forall k \in [m]$,

$$a_{ki} = \begin{cases} 1 \text{ if } g_k \in T_i, \text{ i.e. } g_k \text{ accepts train } T_i \\ 0 \text{ else} \end{cases}$$

This simplification is convenient because the number of qubits, i.e. the number of binary variables, is exactly the number of available trains. Thus, we do not take into account the number of customers, which drastically reduces the number of qubits necessary to describe, and then to solve, an instance. Moreover, we show in the next section that the SCP constraints are easy to integrate as penalty terms in the objective function, and under some assumptions, do not introduce extra qubits for the QUBO formulation.

Notice that we could easily introduce the cost production related to trains by weighting each binary variable with the cost of using a train in the objective function. It would transform the SCP into a weighted SCP. However, the SCP modelization is already simplified to the maximum and thus represents a toy problem to understand what is at stake when solving optimization problems with hybrid metaheuristics, so we omit these weights.

Let us present below a different version of the SCP to get a model a bit closer to the real timetabling problem and to pinpoint the obstacles to scaling this problem to a resolution on current quantum hardware. This model is more realistic because it takes into account the maximum capacity of carriages and does not impose to satisfy all customer groups' demands. Indeed, it avoids using trains for only a small number of customers. Consequently, the introduction of the maximum capacity constraints increases the number of variables because it requires variables to assign groups to trains.

**Extended Bin Packing simplification** For this simplification, we still consider $m$ groups of customers, respectively $n$ available trains, described as for the SCP by the set $G = \{g_1, \dots, g_m\}$, respectively $T = \{T_1, \dots, T_n\}$. Additionally, we suppose that each group contains the same number of customers. The latter assumption does not cause too much loss of generality because one can always define the smallest group as a unit and duplicate groups that are bigger. For each available train $T_i$, $i \in [n]$, we specify

- $p_i$, the benefit of selling tickets to one group for train $T_i$
- $c_i$, the cost of using train $T_i$

We note CMax the maximum number of groups a carriage can accommodate, i.e. the maximum capacity of a carriage divided by the (fixed) number of customers in a group. We define below the problem considered, called Extended Bin Packing.

**Definition 4.1.2** (Extended Bin Packing problem)**.** *The binary decision variables for the Extended Bin Packing problem are of two types. The first one indicates if the train is taken in the timetable:* $\forall i \in [n]$,

$$x_i = \begin{cases} 1 & \text{if } T_i \text{ is taken in the timetable} \\ 0 & \text{else} \end{cases}$$

*The second assigns groups to trains in the timetable:* $\forall i \in [n], \forall j \in [m]$,

$$y_{ij} = \begin{cases} 1 & \text{if group } g_j \text{ takes train } T_i \\ 0 & \text{else} \end{cases}$$

*Notice that we declare a variable $y_{ij}$ if and only if $g_j \in T_i$, namely that train $T_i$ can satisfy group $g_j$. It avoids unnecessary variables and reduces the number of $y$ variables from $nm$ to $q := \sum_{i=1}^{n} |T_i|$. Eventually, the problem is stated as follows, which can be seen as a sort of Bin Packing problem. In our case, each bin has a different capacity, and not every item has to be put*

*in a bin because we are rather looking for the best compromise between the cost of using bins and the reward of putting items into bins. Hence its name, the Extended Bin Packing problem.*

$$\min_{x,y} \quad \sum_{i=1}^{n} c_i x_i - \sum_{i=1}^{n} \sum_{\substack{j\in[m]:\\ g_j\in T_i}} p_i y_{ij} \qquad \text{(Extended-BP)}$$

$$\text{subject to} \quad \sum_{\substack{i\in[n]:\\ g_j\in T_i}} y_{ij} \leq 1, \qquad \forall j \in [m] \qquad \text{(Uni)}$$

$$\sum_{\substack{j\in[m]:\\ g_j\in T_i}} y_{ij} \leq \mathrm{CMax} \cdot x_i, \qquad \forall i \in [n] \qquad \text{(Capa)}$$

$$x_i \in \{0,1\}, \qquad \forall i \in [n]$$

$$y_{ij} \in \{0,1\}, \qquad \forall i \in [n],\ j \in [m] \text{ such that } g_j \in T_i$$

*Constraint* (Uni) *ensures that no customer group takes two trains or more in the timetable, and Constraint* (Capa) *both expresses the limited capacity of a carriage and forbids a group to take a train unused in the timetable.*

## 4.2 Reformulations of simplified problems

In this section, we propose several reformulations of the two simplified problems (SCP) and (Extended-BP) into unconstrained problems to solve them with VQAs later. We remind that the current maturity of gate-based quantum computers (small number of qubits, low connectivity, high error rate of quantum operations, etc.) does not enable to implement algorithms requiring deep depth of consecutive operations and numerous auxiliary qubits additional to the description of the instance on such quantum devices. Hence the use of hybrid quantum-classical heuristics to solve combinatorial problems today. For that, we need to reformulate each of the two simplified problems as Polynomial Unconstrained Binary Optimization (PUBO) problems. In what follows, we present such reformulations, amounting to integrate the constraints as penalty terms in the objective function because the variables are already binary. We also reformulate them as QUBO problems to compare the two reformulations later.

### 4.2.1 Set Cover Problem into PUBO and QUBO

In (SCP), there is only one type of constraint, Constraint (Satisf). For $k \in [m]$, the constraint is equal to a sum of $N_k$ binary variables, where $N_k := \sum_{i=1}^{n} a_{ki}$ represents the number of trains accepted by group $g_k$. Specifically, the constraint is

$$x_{i_1} + \ldots + x_{i_{N_k}} \geq 1 \qquad \text{(Satisf}_k\text{)}$$

where $\{T_{i_1}, \ldots, T_{i_{N_k}}\}$ is the set of trains accepted by $g_k$.

**Proposition 4.2.1** (Penalty term of Constraint (Satisf$_k$))**.** *The penalty function of Con-*

*straint* (Satisf$_k$) *is the following polynomial function of degree* $N_k$*. For* $x \in \{0,1\}^n$*,*

$$pen_k(x) = \prod_{i=i_1}^{i_{N_k}} (1 - x_i)\,. \tag{4.1}$$

*Proof.* One readily verifies that this function penalizes the solutions that violate the constraint (Satisf$_k$). Indeed,

$$\begin{aligned} pen_k(x) &= \begin{cases} 0, & \text{if } \exists i \in \{i_1, \dots, i_{N_k}\} \text{ such that } x_i = 1 \\ 1, & \text{if } x_i = 0 \text{ for all } i \in \{i_1, \dots, i_{N_k}\} \end{cases} \\ &= \begin{cases} 0, & \text{if } x \text{ satisfies (Satisf}_k\text{)} \\ 1, & \text{otherwise} \end{cases} \end{aligned}$$

□

It results in the PUBO reformulation below.

**Proposition 4.2.2** (SCP into PUBO)**.** *The reformulation of* (SCP) *into a PUBO problem is*

$$\min_{x \in \{0,1\}^n} \sum_{i=1}^{n} x_i + \sum_{k \in [m]} \lambda_k pen_k(x) \tag{SCP-PUBO}$$

*where* $\lambda_k \geq 0$ *for any* $k \in [m]$*, and where the penalty function is defined in Proposition 4.2.1*

We recall that any feasible solution of (SCP) has the same loss function value as for (SCP-PUBO), and by extension, the optimal solution(s) is (are) kept identical for $\lambda_k$ large enough.

Henceforth, we work on 2-SCP which we define below.

**Definition 4.2.3** (2-SCP)**.** *We define 2-SCP as the problem* (SCP) *where each element is at most in two subsets, namely* $\sum_{i=1}^{n} a_{ki} \leq 2$ *for each* $k \in [m]$*.*

In other words, working on 2-SCP supposes that each customer group accepts to take at most two different trains. This assumption is reasonable because the nominal railway timetabling problem is dedicated to high-speed trains, so two consecutive available trains from the same origin and the same destination leave at a relatively long time interval, minimizing the number of customers that would accept to take three or more trains. We choose to work on 2-SCP because the resulting penalty functions (4.1) are linear or quadratic (because we assume that $N_k \in \{1, 2\}$ in 2-SCP), meaning that (SCP-PUBO) is, in reality, a QUBO problem and thus is not too heavy to handle on current computers. Moreover, as mentioned earlier, because a generic QUBO is already NP-hard to solve, the interest of studying empirically such problems is still valid.

2-SCP features two types of constraints. Those stemming from customer groups accepting only one train. We note $I_1$ the indexes of these trains, namely $I_1 := \{i \in [n] : \exists k \in [m] \text{ such that } g_k \in T_i \text{ and } g_k \notin \cup_{j \neq i} T_j\}$. For $i \in I_1$, the constraint is

$$x_i \geq 1\,.$$

The other type of constraint stems from customer groups accepting exactly two trains. We note $I_2$ the couples of indexes of these trains, namely $I_2 := \{(i,j) \in [n]^2 : \exists k \in [m] \text{ such that } g_k \in$

$T_i \cap T_j$ and $g_k \notin \cup_{l \neq i,j} T_l\}$. For $(i,j) \in I_2$, the constraint is

$$x_i + x_j \geq 1\,.$$

Thus, the QUBO formulation of 2-SCP is the following.

**Proposition 4.2.4** (2-SCP into QUBO)**.** *The reformulation of 2-SCP as a QUBO problem is*

$$\min_{x \in \{0,1\}^n} \sum_{i=1}^{n} x_i + \lambda_1 \sum_{i \in I_1} (1 - x_i) + \lambda_2 \sum_{(i,j) \in I_2} (1 - x_i)(1 - x_j) \tag{2-SCP-QUBO}$$

*for* $\lambda_1, \lambda_2 \in \mathbb{R}_+^*$.

Notice that we set equal all penalty coefficients related to the same type of constraint. Indeed, intuitively, there is no reason to weight one constraint more than another if they are of the same type and the trains' costs are assumed to be identical.

### 4.2.2 Extended Bin Packing problem into PUBO and QUBO

To transform (Extended-BP) into a PUBO problem, we need to integrate two different types of constraints. The penalty functions we propose next have the particularity to be equal to 1 if the constraint is violated and 0 otherwise. The choice of such binary functions is motivated by the fact that knowing the cost of the violation (always equal to 1 here) is convenient for controlling the overall violation cost as we will see in the last section of this chapter.

In what follows, we provide penalty functions (of binary values) of generic equality and inequality constraints, where the variable-dependant term is the sum of binary variables, that encompass the case of the constraints ($\text{Uni}_j$) and ($\text{Capa}_i$). We express the corresponding penalty terms for these two specific constraints after. Let us begin with the penalty term for equality constraints.

**Property 4.2.5.** *Let us consider the constraint, for* $n \in \mathbb{N}$*,*

$$\sum_{i=1}^{n} x_i = c\,. \tag{Eq}$$

*For* $c \in \mathbb{N}^*$*, the penalty function is, for* $x \in \{0,1\}^n$*,*

$$\pi_c^{eq}(x) = 1 + \sum_{k=c}^{n} (-1)^{k-c+1} \binom{k}{c} \sum_{i=\{i_1,\ldots,i_k\} \in I_k^n} x_{i_1} \ldots x_{i_k}\,,$$

*where* $I_k^n$ *denotes all the sets of* $k$ *elements in* $[n]$*. For the specific case of* $c = 0$*, the penalty function is*

$$\pi_0^{eq}(x) = \sum_{k=1}^{n} (-1)^{k+1} \sum_{i \in I_k^n} x_{i_1} \ldots x_{i_k}\,.$$

*Proof.* Let $x \in \{0,1\}^n$. Let us begin with the case $c = 0$.

- If $x$ satisfies (Eq), then $\sum_{i=1}^n x_i = 0$ by definition, and it directly results that $\pi_0^{\text{eq}}(x) = 0$.
- If $x$ violates (Eq), then we note $\alpha := \sum_{i=1}^n x_i$. By definition of the violation, $\alpha \in [\![1, n]\!]$.

Thus, because for $k > \alpha$, any product of $k$ variables is equal to 0,

$$\begin{aligned}
\pi_0^{\mathrm{eq}}(x) &= \sum_{k=1}^{\alpha}(-1)^{k+1}\sum_{i\in I_k^\alpha} x_{i_1}\dots x_{i_k} \\
&= \sum_{k=1}^{\alpha}(-1)^{k+1}\binom{\alpha}{k} \\
&= -\left(\sum_{k=0}^{\alpha}(-1)^{k}\binom{\alpha}{k} - 1\right) \\
&= 1 - \sum_{k=0}^{\alpha}(-1)^{k}1^{\alpha-k}\binom{\alpha}{k} = 1\,. \quad \text{(Newton binomial formula)}
\end{aligned}$$

Let us next consider the case $c \in \mathbb{N}^*$.

- If $x$ satisfies (Eq), then it exists $c$ variables equal to 1. Let us refer to them as $\hat{x}_1, \dots, \hat{x}_c$. Thus, $\sum_{i\in I_c^n} x_{i_1}\dots x_{i_c} = \hat{x}_1\dots\hat{x}_c = 1$, and $\sum_{i\in I_k^n} x_{i_1}\dots x_{i_k} = 0$ for $k > c$. It results that

$$\pi_c^{\mathrm{eq}}(x) = 1 + (-1)^{c-c+1}\binom{c}{c}\cdot 1 = 1 - 1 = 0\,.$$

- If $x$ violates (Eq), then by definition $\sum_{i=1}^n = \alpha \neq c$:
  - If $\alpha < c$, thus $\sum_{i\in I_k^n} x_{i_1}\dots x_{i_k} = 0$ for any $k \geq c$, leading to $\pi_c^{\mathrm{eq}}(x) = 1 - 0 = 1$.
  - If $\alpha > c$, thus $\sum_{i\in I_k^n} x_{i_1}\dots x_{i_k} = \binom{\alpha}{k}$ for any $c \leq k \leq \alpha$, and $\sum_{i\in I_k^n} x_{i_1}\dots x_{i_k} = 0$ for any $k > \alpha$. It results that

$$\pi_c^{\mathrm{eq}}(x) = 1 - \sum_{k=c}^{\alpha}(-1)^{k-c+1}\binom{k}{c}\binom{\alpha}{k},$$

    where we can show by manipulating factorials that

$$\binom{k}{c}\binom{\alpha}{k} = \binom{\alpha}{c}\binom{\alpha-c}{k-c}.$$

    Thus,

$$\begin{aligned}
\pi_c^{\mathrm{eq}}(x) &= 1 - \sum_{k=c}^{\alpha}(-1)^{k-c+1}\binom{\alpha}{c}\binom{\alpha-c}{k-c} \\
&= 1 + \binom{\alpha}{c}\sum_{k=0}^{\alpha-c}(-1)^{k}\binom{\alpha-c}{k} \\
&= 1 + \binom{\alpha}{c}(1-1)^{\alpha-c} = 1\,.
\end{aligned}$$

□

**Property 4.2.6.** *We consider the constraint, for $n \in \mathbb{N}$ and $c \in \mathbb{N}$,*

$$\sum_{i=1}^{n} x_i \leq c\,. \tag{Inf}$$

*The associated penalty function is, for $x \in \{0,1\}^n$,*

$$\pi_c^{inf}(x) = \sum_{k=c+1}^{n} (-1)^{k-c+1} \binom{k-1}{c} \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k}\,.$$

*Proof.* Let $x \in \{0,1\}^n$.

- If $x$ satisfies (Inf), then $\sum_{i \in I_k^n} x_{i_1} \dots x_{i_1} = 0$ for any $k > c$. Thus, $\pi_c^{\inf}(x) = 0$.
- If $x$ violates (Inf), let us note $\sum_{i=1}^{n} x_i = \alpha > c$. Thus,

$$\sum_{i \in I_k^n} x_{i_1} \dots x_{i_k} = \begin{cases} \binom{\alpha}{k} & \text{for } k \in [\![c+1, \alpha]\!] \\ 0 & \text{for } k > \alpha \end{cases}$$

  It results that

$$\pi_c^{\inf}(x) = \sum_{c+1}^{\alpha} (-1)^{k-c+1} \binom{k-1}{c} \binom{\alpha}{k}\,.$$

  Next, we prove by recurrence over $c \in [\![0, \alpha-1]\!]$ the proposition

$$\mathcal{P}(c) : \text{“}\pi_c^{\inf}(x) = 1, \text{ for all } x \in \{0,1\}^n \text{ violating (Inf)”}$$

  *Initialization:* $\mathcal{P}(\alpha-1)$ is True. Indeed, for $x$ violating (Inf), $\pi_{\alpha-1}^{\inf}(x) = (-1)^{\alpha-(\alpha-1)+1}\binom{\alpha-1}{\alpha-1}\binom{\alpha}{\alpha} = 1$.

  *Recurrence:* Let $c \in [\![1, \alpha-1]\!]$, and let us assume that $\mathcal{P}(c)$ is True. Let us show that $\mathcal{P}(c-1)$ is also True. For $x$ violating (Inf),

$$\begin{aligned}
\pi_{c-1}^{\inf}(x) &= \sum_{k=c}^{\alpha} (-1)^{k-c} \binom{k-1}{c-1} \binom{\alpha}{k} \\
&= (-1)^{c-c} \binom{c-1}{c-1} \binom{\alpha}{c} + \sum_{k=c+1}^{\alpha} (-1)^{k-c} \binom{k-1}{c-1} \binom{\alpha}{k} \\
&= \binom{\alpha}{c} - \sum_{k=c+1}^{\alpha} (-1)^{k-c+1} \left( \binom{k}{c} - \binom{k-1}{c} \right) \binom{\alpha}{k} \quad \text{(Pascal's triangle)} \\
&= \binom{\alpha}{c} - \sum_{k=c+1}^{\alpha} (-1)^{k-c+1} \binom{k}{c} \binom{\alpha}{k} + \underbrace{\sum_{k=c+1}^{\alpha} (-1)^{k-c+1} \binom{k-1}{c} \binom{\alpha}{k}}_{\pi_c^{\inf}(x)} \\
&= \binom{\alpha}{c} - \sum_{k=c+1}^{\alpha} (-1)^{k-c+1} \binom{k}{c} \binom{\alpha}{k} + 1\,. \quad \text{(Recurrence hypothesis)}
\end{aligned}$$

Identically to the proof of the penalty term of constraint (Eq) for the case $c \in \mathbb{N}^*$, we can show that

$$\sum_{k=c+1}^{\alpha} (-1)^{k-c+1} \binom{k}{c} \binom{\alpha}{k} = \binom{\alpha}{c}.$$

Thus, it results that $\pi_{c-1}^{\inf}(x) = \binom{\alpha}{c} - \binom{\alpha}{c} + 1 = 1$.

□

Next, we define a penalty term for inequality constraints. Property 4.2.6 deals with the inferiority case whereas Property 4.2.7 tackles the superiority case.

**Property 4.2.7.** *We consider the constraint, for $n \in \mathbb{N}$ and $c \in \mathbb{N}^*$,*

$$\sum_{i=1}^{n} x_i \geq c. \tag{Sup}$$

*The associated penalty function is, for $x \in \{0,1\}^n$,*

$$\pi_c^{sup}(x) = 1 + \sum_{k=c}^{n} (-1)^{k-c+1} \binom{k-1}{c-1} \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k}.$$

*Proof.* Let $x \in \{0,1\}^n$. It is sufficient to show that $\pi_c^{\sup}(x) = \pi_c^{\text{eq}}(x) - \pi_c^{\inf}(x)$, for $c \in \mathbb{N}^*$. Indeed, if we note $\alpha := \sum_{i=1}^{n} x_i$, we have the following results.

- If $x$ satisfies (Sup) (meaning that $\alpha \geq c$):
  - If $\alpha = c$: $\pi_c^{\text{eq}}(x) = \pi_c^{\inf}(x) = 0$, thus $\pi_c^{\sup}(x) = 0$
  - If $\alpha > c$: $\pi_c^{\text{eq}}(x) = \pi_c^{\inf}(x) = 1$, thus $\pi_c^{\sup}(x) = 0$
- If $x$ violates (Sup) (meaning that $\alpha < c$), we have $\pi_c^{\text{eq}}(x) = 1$ and $\pi_c^{\inf}(x) = 0$, leading to $\pi_c^{\sup}(x) = 1$

Thus, it remains to prove that $\pi_c^{\sup}(x) = \pi_c^{\text{eq}}(x) - \pi_c^{\inf}(x)$.

$$\begin{aligned}
\pi_c^{\text{eq}}(x) - \pi_c^{\inf}(x) &= 1 + \sum_{k=c}^{n} (-1)^{k-c+1} \binom{k}{c} \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k} - \sum_{k=c+1}^{n} (-1)^{k-c+1} \binom{k-1}{c} \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k} \\
&= 1 + (-1)^{c-c+1} \binom{c}{c} \sum_{i \in I_c^n} x_{i_1} \dots x_{i_k} + \sum_{k=c+1}^{n} (-1)^{k-c+1} \left( \binom{k}{c} - \binom{k-1}{c} \right) \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k} \\
&= 1 - (-1)^{c-c+1} \binom{c-1}{c-1} \sum_{i \in I_c^n} x_{i_1} \dots x_{i_k} + \sum_{k=c+1}^{n} (-1)^{k-c+1} \binom{k-1}{c-1} \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k} \\
&= 1 + \sum_{k=c}^{n} (-1)^{k-c+1} \binom{k-1}{c-1} \sum_{i \in I_k^n} x_{i_1} \dots x_{i_k} \\
&= \pi_c^{\sup}(x).
\end{aligned}$$

□

It results from the previous properties the following penalty terms for the Extended Bin Packing problem. The first constraint of (Extended-BP), expressing that a group takes at most one train, is the following: for all $j \in [m]$,

$$\sum_{\substack{i\in[n]:\\ g_j\in T_i}} y_{ij} \leq 1\,. \tag{Uni$_j$}$$

It follows the expression of the penalty term.

**Proposition 4.2.8** (Penalty term for Constraint (Uni$_j$))**.** *Let $j \in [m]$ and let us note $G_j$ the set of indexes of trains accepted by group $g_j$. The function, for $x \in \{0,1\}^n$, $y \in \{0,1\}^q$,*

$$penUni_j(x,y) = \sum_{k=2}^{|G_j|} (-1)^k (k-1) \sum_{\substack{I=(i_1,\dots,i_k)\subseteq G_j:\\ |I|=k}} y_{i_1,j}\dots y_{i_k,j}$$

*is a penalty term for Constraint* (Uni$_j$)*.*

*Proof.* Use Property 4.2.6 for the binary variables $y_{ij}$ such that $i \in G_j$, and for the constant $c = 1$. □

The second type of constraint of (Extended-BP), the capacity constraint, is the following: for all $i \in [n]$,

$$\sum_{\substack{j\in[m]:\\ g_j\in T_i}} y_{ij} \leq \mathrm{CMax}\cdot x_i\,. \tag{Capa$_i$}$$

Its penalty function is described below.

**Proposition 4.2.9** (Penalty term for Constraint (Capa$_i$))**.** *Let $i \in [n]$. The following function is a penalty term for Constraint* (Capa$_i$)*. For $x \in \{0,1\}^n$, $y \in \{0,1\}^q$,*

$$penCapa_i(x,y) = (1-x_i)\cdot pen_0(x,y) + x_i \cdot pen_1(x,y)\,,$$

*where*

$$pen_0(x,y) = \sum_{k=1}^{|\{j:i\in G_j\}|} (-1)^{k+1} \sum_{\substack{J\subseteq\{j:i\in G_j\}:\\ |J|=k}} y_{i,j_1}\dots y_{i,j_k}$$

*and*

$$pen_1(x,y) = \sum_{k=\mathrm{CMax}+1}^{|\{j:i\in G_j\}|} (-1)^{k-\mathrm{CMax}+1} \binom{k-1}{\mathrm{CMax}} \sum_{\substack{J\subseteq\{j:i\in G_j\}:\\ |J|=k}} y_{i,j_1}\dots y_{i,j_k}\,.$$

*Proof.* We distinguish two cases.

- If $x_i = 0$, then we use Property 4.2.5 for the binary variables $y_{ij}$ such that $j \in \{j : i \in G_j\}$, and for the constant $c = 0$, which provides the term $pen_0(x,y)$.
- If $x_i = 1$, then we use Property 4.2.6 for the binary variables $y_{ij}$ such that $j \in \{j : i \in G_j\}$, and for the constant $c = \mathrm{CMax}$, which provides the term $pen_1(x,y)$.

Eventually, we can add the two term as follows, each one appearing when the condition on $x_i$ is satisfied, to obtain the final penalty function: $(1 - x_i) \cdot pen_0(x, y) + x_i \cdot pen_1(x, y)$. □

Notice that we do not use Property 4.2.7 for the constraints of our problem, but we presented it to provide all the generic cases. It results from the two propositions above the reformulation of our simplified problem (Extended-BP) into a PUBO problem.

**Proposition 4.2.10** (Extenced-BP into PUBO)**.** *The reformulation of* (Extended-BP) *as a PUBO problem is:*

$$\min_{\substack{x\in\{0,1\}^n \\ y\in\{0,1\}^q}} \sum_{i=1}^{n} c_i x_i - \sum_{i=1}^{n} \sum_{\substack{j\in[m]: \\ g_j\in T_i}} p_i y_{ij} \qquad \text{(Extended-BP-PUBO)}$$
$$+ \sum_{j=1}^{m} \lambda_{u,j} penUni_j(x, y) + \sum_{i=1}^{n} \lambda_{c,i} penCapa_i(x, y)\,,$$

*where $\lambda_{u,j}, \lambda_{c,i} \in \mathbb{R}_+^*$ for all $j \in [m], i \in [n]$ are the penalty coefficients, and where the penalty functions are defined in Propositions 4.2.8 and 4.2.9.*

Next, we present a reformulation into a QUBO problem and show that it requires auxiliary qubits, and thus weakens the interest of such a formulation for VQAs.

**Proposition 4.2.11** (Quadratic penalty terms of Constraints ($\text{Uni}_j$) and ($\text{Capa}_i$))**.** *The penalty term for the first Constraint* ($\text{Uni}_j$) *is, for $j \in [m]$,*

$$penUniQubo_j(x, y, s) = \left( \sum_{i\in G_j} y_{ij} + s_j - 1 \right)^2 ,$$

*where $s_j$ is an additional binary variable. The penalty term for the second constraint* ($\text{Capa}_i$) *is, for $i \in [n]$,*

$$penCapaQubo_i(x, y, r) = \left( \sum_{j:i\in G_j} y_{ij} - \text{CMax} \cdot x_i + r_i \right)^2 ,$$

*where $r_i \in [\![0, \text{CMax}]\!]$ is an additional integer variable.*

In the binary model, we replace $r_i$ by its binary expression $r_i^{bin} := \sum_{l=0}^{\lfloor \log_2(\text{CMax}) \rfloor} 2^k \cdot r_{i,k}^{bin}$, for $r_{i,k}^{bin} \in \{0, 1\}$. In total, the penalty constraints require roughly $(m + n \cdot \log_2(\text{CMax}))$ additional binary variables to the initial Extended Bin Packing problem to express the following QUBO formulation.

**Proposition 4.2.12** (Extended-BP into QUBO)**.** *The reformulation of* (Extended-BP) *as a QUBO problem is*

$$\min_{\substack{x\in\{0,1\}^n, \\ y\in\{0,1\}^q \\ s\in\{0,1\}^m, \\ r^{bin}\in\{0,1\}^{n\cdot\log_2(\text{CMax})}}} \sum_{i=1}^{n} c_i x_i - \sum_{i=1}^{n} \sum_{j\in[m]:g_j\in T_i} p_i y_{ij} \qquad \text{(Extended-BP-QUBO)}$$

$$+\sum_{j=1}^{m}\lambda_{u,j}penUniQubo_j(x,y,s)+\sum_{i=1}^{n}\lambda_{c,i}penCapaQubo_i(x,y,r^{bin})\,,$$

*where* $\lambda_{u,j},\lambda_{c,i}\in\mathbb{R}_+^*$ *for all* $j\in[m],i\in[n]$ *are the penalty coefficients, and where the penalty functions are defined in Proposition 4.2.11.*

Not only does the QUBO formulation require additional qubits but it also has penalty terms that take integer values (and not binary values such as in the PUBO formulation). These two drawbacks are discussed in Subsection 4.3.2, together with numerical results of the resolutions with QAOA of several instances of 2-SCP and Extended Bin Packing problem.

## 4.3 Resolution with QAOA

### 4.3.1 Influence of QAOA parameters

In this subsection, we study the impact of some parameters of QAOA on its performance. First, we identify some limitations due to the choice of QAOA circuit. Second, we examine the influence of depth on the latter circuit. Notice that we do not provide an exhaustive description of the limits and advantages of QAOA but we rather aim to point out some trends to understand better the performances of QAOA to solve problems with current quantum hardware, namely, on small instances.

#### 4.3.1.1 QAOA circuit covering

In what follows, we discuss the expression of the parametrized quantum circuit of QAOA and its impact on the sharpness of covering the quantum space. We showed in Section 3.1 that the choice of the quantum circuit to define a VQA is important. Indeed, it requires finding a trade-off between the number of parameters to be optimized classically and the sharpness to produce any quantum state and be able to set a large weight on the basis state(s) corresponding to the optimal solution(s). Hereafter, we exhibit some results showing the limits of the design of QAOA's circuit.

Let us begin with an example. We consider the following minimization problem

$$\min_{x_1,x_2\in\{0,1\}} -10x_1+5x_2-3x_1x_2$$

for which the optimal solution is $(x_1=1,x_2=0)$ for a loss value equal to -10. We show in Figure 4.1 the energy landscape to be optimized by the classical optimizer by running QAOA with depth $p=1$. For that, we compute the values of the guiding function $g$, which is the average function, of each quantum state produced by QAOA circuit for depth $p=1$ for each couple of parameters $(\gamma,\beta)\in[0,2\pi]^2$ (with granularity 0.1).

We compare it with the landscape Figure 4.2 obtained when running VQA with the circuit

$$U(\theta_1,\theta_2)=R_Y(\theta_1)\otimes R_Y(\theta_2)\,,\quad\text{for }(\theta_1,\theta_2)\in[0,2\pi]^2\,,$$

as in Example 6. Notice that we evaluate the same guiding function $g$ as for QAOA, which is the expectation value of the costs' solutions.

The energy landscape for QAOA is not *stable*, explained by the fact that the guiding function $g$ is essentially a sum of products of sinusoidal functions of $\gamma$ and $\beta$, and then indicates a rather difficult optimization over $\gamma$ and $\beta$. Moreover, we notice that the optimal value -10 is never reached because the circuit cannot produce the pure basis state $|10\rangle$ representing the optimal

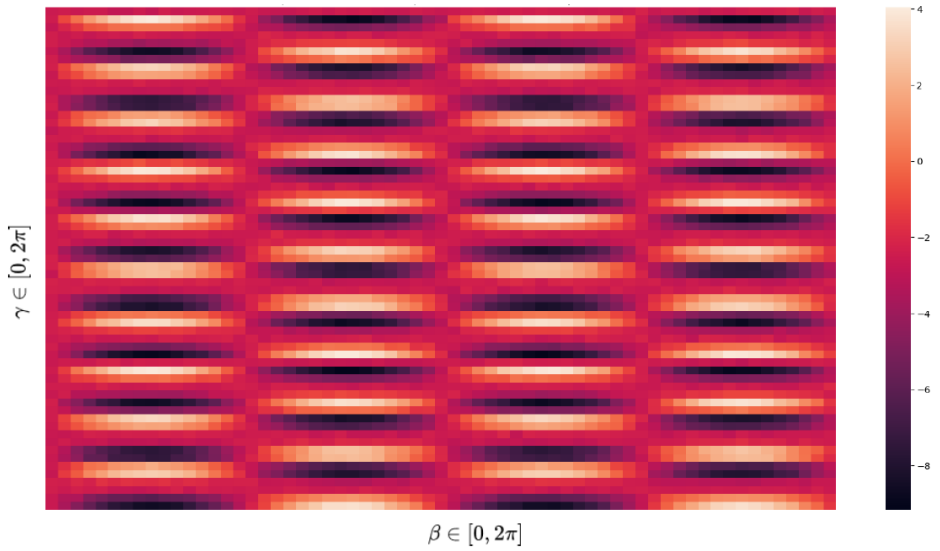


Figure 4.1: Guiding function values of quantum states produced by QQAOA circuit for depth $p = 1$.

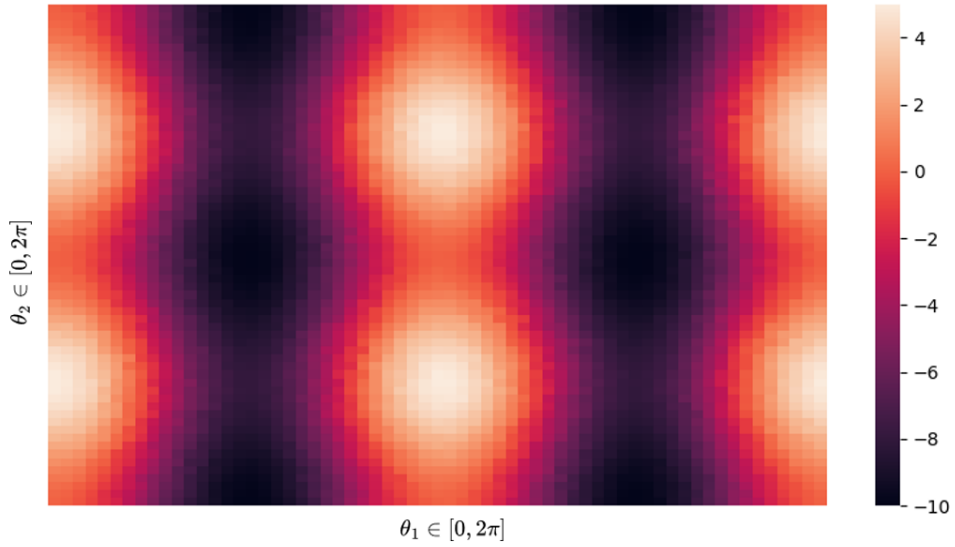


Figure 4.2: Guiding function values of quantum states produced by circuit $U(\theta_1, \theta_2)$.

solution $(1, 0)$. For the chosen granularity 0.1, the produced quantum state with the largest probability on the optimal state is a state with a probability equal to 0.863 for $|10\rangle$. On the contrary, the landscape for VQA with circuit $U(\theta_1, \theta_2)$ is smoother and reaches the optimal state with probability 1. This small example focuses on the limits of QAOA circuit for depth $p = 1$ and the necessity to increase the depth to sharpen the covering of quantum states.

Next, we illustrate in Figure 4.3 this phenomenon on 2-SCP (Definition 4.2.3). For a given number of available trains (which is exactly the size of the instance), we generate 20 random

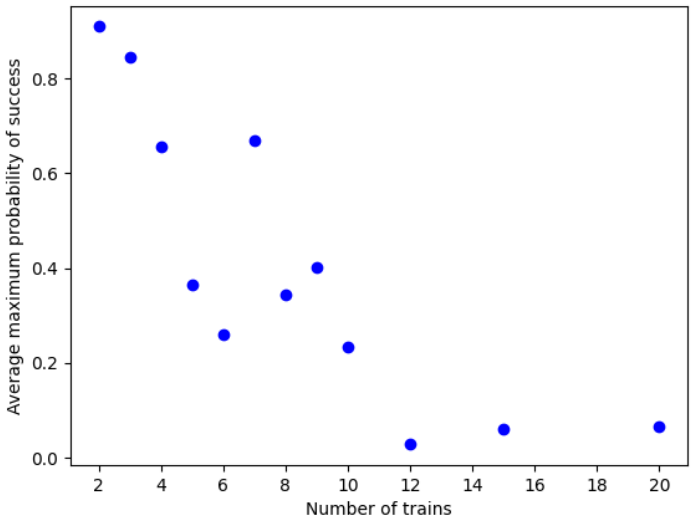


Figure 4.3: Average maximum probability of the optimal solution for quantum states produced by QAOA circuit of depth $p = 1$ over 20 random 2-SCP instances.

2-SCP instances and approximate the average maximum probability on the optimal solution(s) when sampling the energy landscape for QAOA with depth $p = 1$. For that, we proceed as follows. For each instance, we vary the parameters $(\gamma, \beta)$ in $[0, 2\pi]^2$ (with granularity 0.1). For each couple of parameters, we sample with 1024 shots (on the 32-qubit QASM simulator of IBM) the quantum state resulting from the execution of the parametrized quantum circuit. We store the maximum probability of success of the optimal solution(s) over all these couples. Eventually, we compute the average maximum probability of success for all 2-SCP instances with the same number of trains.

We notice that the trend of the evolution of the maximum probability of success is to decrease drastically when the number of trains increases. When reaching instances of 20 trains, the average maximum probability of the output quantum state is no more than 0.006. In other words, if we suppose that the classical optimizer finds the optimal parameters $(\gamma^*, \beta^*)$ (still with granularity 0.1), the optimal value of the 2-SCP will represent only 0.6% of the measurement outcomes of the corresponding quantum state. Even if the computation of each maximum probability of success is an estimation, due to the discretization of real parameters $(\gamma, \beta)$ on the one hand, and the limited number of shots for the sampling on the other hand, these experiments aims at pointing out a trend.

A legitimate question arising after these first analyses is whether a larger depth would improve the results of QAOA. On the one hand, the maximum probability of the optimal solution(s) increases with the depth because any quantum state for the circuit of depth $p$ can be produced by the circuit of depth $p + 1$ by setting the two new parameters $\gamma_{p+1}$ and $\beta_{p+1}$ to zero. On the other hand, the energy landscape gets more complicated with a bigger number of parameters to optimize classically. Thus, we investigate the influence of the depth next.

#### 4.3.1.2 Depth of QAOA circuit

The previous analysis led us to investigate the influence of the depth $p$ of QAOA circuit. For all the numerical results that follow, we implemented our own QAOA on Qiskit (IBM), where the classical optimizer is chosen to be COBYLA which is provided by the Qiskit Library, and the execution of the quantum circuit is done on the QASM simulator of 32 qubits. For this work, we chose to run the quantum circuit on a simulator instead of on a real quantum device because we focus on depth influence, and more generally on QAOA performances. Thus, we do not want the noise to interfere with the analysis. Indeed, the current error rate of quantum gates is too high to have interpretable results on QAOA performances. If we apply $N$ gates in a row that have an average error rate of $\tau$, the error on one qubit of the output quantum state is $1-(1-\tau)^N$. To this day, the one-qubit gates on IBM quantum machines have a median error rate of $\tau \approx 2.3 \times 10^{-4}$, which means that applying 100 gates generates around 3% of error. For two-qubit gates, the error rate goes up to $\tau \approx 7.7 \times 10^{-3}$ leading to 54% of error for the same number of gates.

Let us present the results of depth influence on Figure 4.4. For each number of trains between 2 and 8, and for each depth between 1 and 5, we generate 15 random 2-SCP instances and solve each of them 10 times with QAOA to get an average probability of the optimal solution in the output quantum state of QAOA (cross line). In other words, the success probability of QAOA represents the average proportion of the optimal state(s) in the output of QAOA. Notice that, henceforth, we call *output quantum state of QAOA* the quantum state produced by the parametrized circuit for the best parameters found by the classical optimizer. In parallel, for each 2-SCP instance, we compute the proportion of the optimal solution(s) in the search space and take the average proportion over all the instances for a given number of trains (dot line). Thus, this proportion represents the average probability of success of picking randomly one state in the search space (one random sampling).

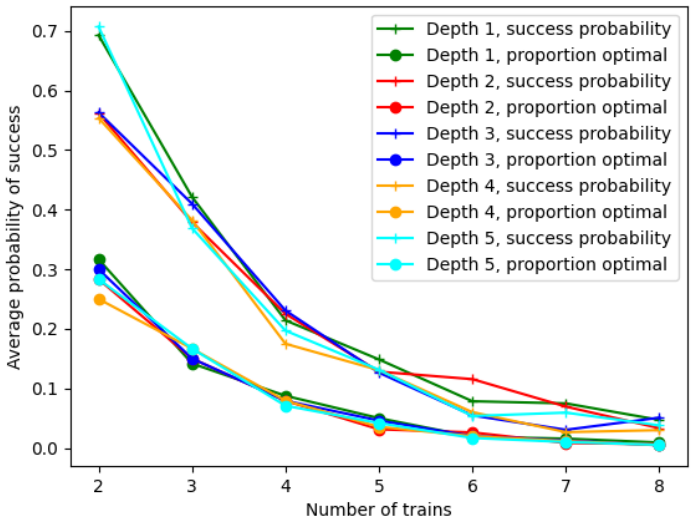


Figure 4.4: Average probability of the optimal solution of QAOA's output quantum state output over 15 random 2-SCP instances and 10 runs of QAOA (cross line). Comparison with one random sampling (dot line).

We observe two things. First, the depth does not seem to change the probability of success of QAOA. We could expect that increasing the depth would improve the performances of QAOA

because the covering of the quantum state space gets more precise, but the results show that the classical optimization over the parameters gets harder. Second, QAOA outperforms one random sampling. However, we note that the exponential decrease of the maximum probability of the optimal solution, illustrated in Figure 4.3, makes the performances of QAOA closer to one random sampling performance as the size of the problem increases. But, still, because the success probability of QAOA is always larger than the proportion of optimal solutions, sampling the output quantum state of QAOA and sampling randomly the search space with the same number of samples gives the advantage to QAOA. Thus, it shows that QAOA *deform* the search space in favor of the optimal solution(s). Despite this, the comparison reaches its limit here because the resources of QAOA to achieve such a *distortion* are huge (classical optimization + execution of quantum circuits + sampling of the output of the circuit) compared to a random algorithm.

### 4.3.2 Numerical results

#### 4.3.2.1 2-SCP on SNCF real instances

We solve the Set Cover Problem on two real instances of SNCF, using the (2-SCP-QUBO) reformulation. The instances consist of solving the timetabling railway problem considering four stations (Paris Gare de Lyon, Lyon Part-Dieu, Lyon Perrache, and Saint-Etienne) on the period of a regular Monday. This perimeter is marked out in the reticular Figure 4.15 by the blue box. The first instance, called $\mathcal{I}_{22}$, considers Monday between 8 a.m. and 12 a.m., whereas the second instance, called $\mathcal{I}_{32}$, extends the period by considering Monday between 8 a.m. and 2 p.m. For the simplified model (SCP), Instance $\mathcal{I}_{22}$, respectively Instance $\mathcal{I}_{32}$, represents an instance of size 22 (22 trains available trains), respectively of size 32 (32 trains available trains), hence their name. For the larger one, $\mathcal{I}_{32}$, this amounts to around 4000 customers gathering into 40 customer groups. Note that they represent tiny instances compared to those that must solve SNCF, both in terms of duration and perimeter. The original classical method is capable of solving instances for a whole day on region-like perimeters, even if not at optimality, and the ideal goal would be to solve the problem on the whole French territory over several months.

Let us begin with the smaller instance $\mathcal{I}_{22}$. In Figures 4.5 and 4.6, we display the results of solving this instance with QAOA, where the quantum part is simulated on the 32-qubit simulator of IBM (QASM simulator). Specifically, we consider the (2-SCP-QUBO) formulation for which we set the penalty coefficients equal $\lambda_1 = \lambda_2 = \lambda$, and vary the values of the penalty coefficients. For each value of $\lambda$, we run 10 times QAOA (with 3000 shots to sample quantum states). On the one hand, we compute the average cost of the solution returned (Figure 4.5). On the other hand, we count the number of runs (among the 10 runs) for which the solution returned was optimal (Figure 4.6). For this instance, the optimal solution is composed of 11 trains. We plot the results of the same experiments for Instance $\mathcal{I}_{32}$ in Figures 4.7 and 4.8, for which the optimal solution is composed of 17 trains.

We observe that for both instances, the value of the penalty coefficients in the (2-SCP-QUBO) reformulation does not seem to impact too much the quality of the solution returned by QAOA, even if sometimes the optimal solution is not found in 10 runs. Regarding the depth, even if the impact is low too, QAOA with a circuit of depth 1 seems to find more often the optimal solution than with depth 3. Specifically, for Instance $\mathcal{I}_{22}$ and for $\lambda \in \{50, 65\}$, the optimal solution is found every other time.

Next, for depth 1 and $\lambda = 50$, we illustrate the shape of the output quantum state of QAOA, namely the distortion that QAOA produces over the uniform distribution in the search space. In Figure 4.9, we display the Cumulative Distribution Functions (CFD) of the samples of output

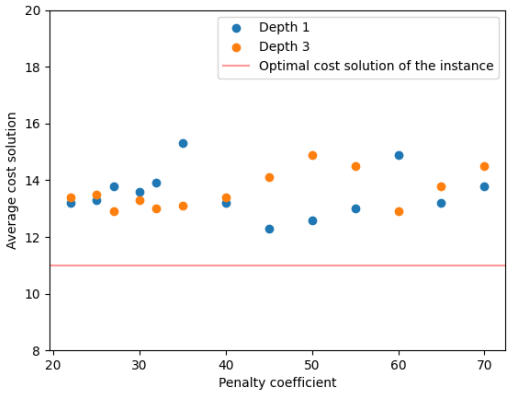


Figure 4.5: Average cost solution of QAOA over 10 runs for Instance $\mathcal{I}_{22}$, varying the depth and the penalty coefficient values.

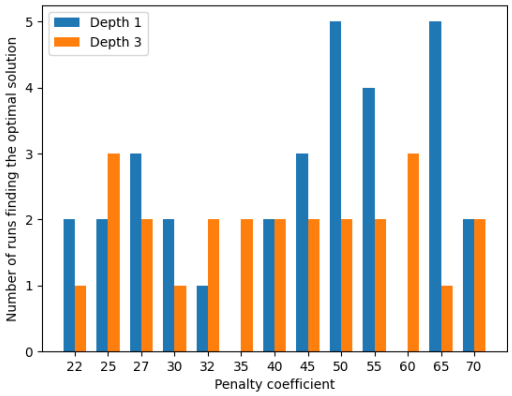


Figure 4.6: Number of runs, among 10 runs, for which QAOA finds the optimal solution for Instance $\mathcal{I}_{22}$, varying the depth and the penalty coefficient values.

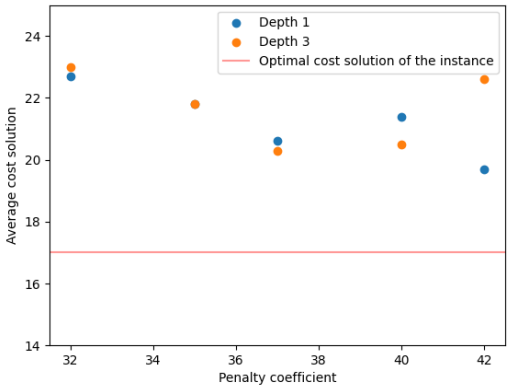


Figure 4.7: Average cost solution of QAOA over 10 runs for Instance $\mathcal{I}_{32}$, varying the depth and the penalty coefficient values.

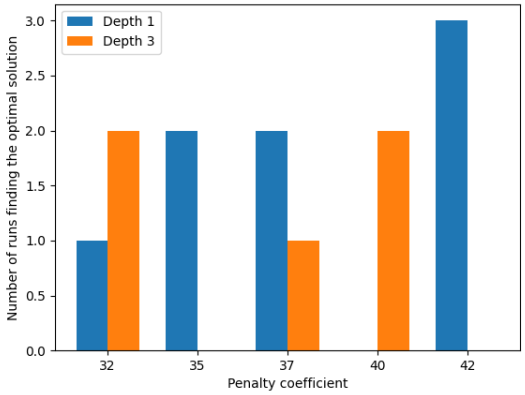


Figure 4.8: Number of runs, among 10 runs, for which QAOA finds the optimal solution for Instance $\mathcal{I}_{32}$, varying the depth and the penalty coefficient values.

quantum states resulting from 2 runs of QAOA (orange and purple lines) as functions of the ratio between the solution found and the optimal solution. We compare them with the CFD representing a uniform state over the search space $\{0,1\}^{22}$ (gray line). The latter represents the CFD for a result with no optimization at all, showing the *energy landscape* associated with the QUBO formulation of Instance $\mathcal{I}_{22}$.

We note that the distortion can be more efficient for some runs (purple line) than others (orange line), even if each case confirms that QAOA outperforms one random sampling as previously mentioned. Notice that the zoom of the previous figure displayed in Figure 4.10 shows that a huge distortion does not always imply finding the optimal solution. Indeed, in this example, the first run of QAOA (orange) finds the optimal solution whereas the second does not (purple).

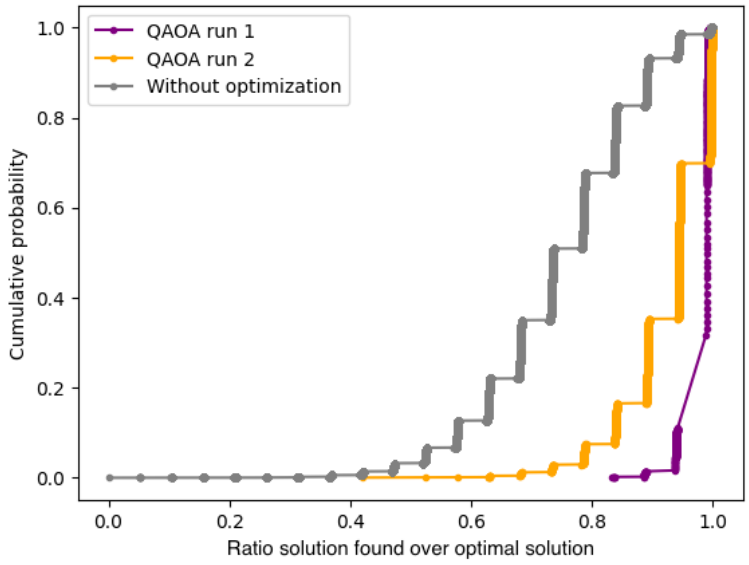


Figure 4.9: Cumulative Distribution Functions for 2 runs of QAOA ($p = 1$, $\lambda = 50$) and without optimization.

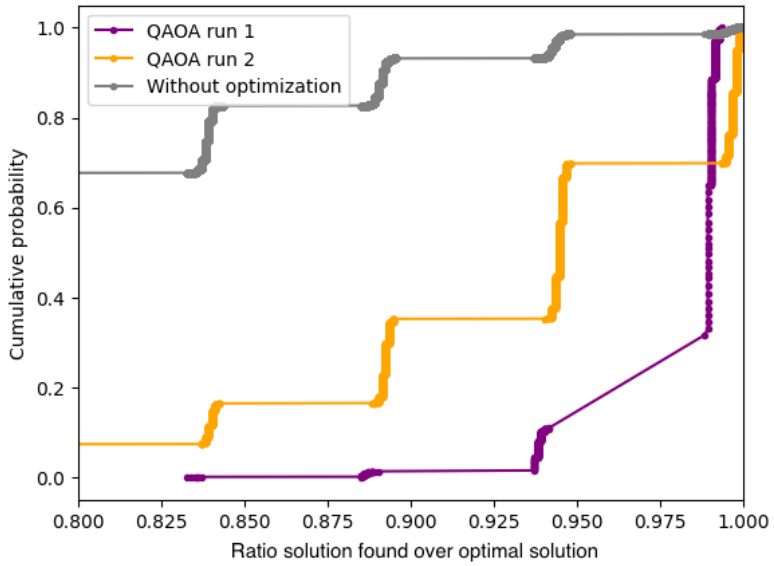


Figure 4.10: Zoom on Figure 4.9 for ratio in $[0.9, 1]$.

#### 4.3.2.2 Different reformulations for Extended Bin Packing

In the previous subsection, we studied the (SCP) simplified formulation. Hereafter, we focus on the other simplified formulation, the (Extended-BP) formulation. Let us illustrate the importance of the choice of the penalty functions with the resolution of this problem. Specifically, we compare

the results of the (Extended-BP-QUBO) and the (Extended-BP-PUBO) reformulations on small instances.

First, we present in Figure 4.12 the results of solving Instance A (Figure 4.11) with QAOA with the QUBO and PUBO formulations. For both formulations, we set $\lambda_u = \lambda_c = 12$, respectively

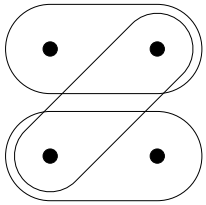

Figure 4.11: Instance A

Instance A: $n = 3$ trains (sets) and $m = 4$ groups (dots), for

- $c_i = 1\,, \forall i \in [n]$
- $p_i = 1\,, \forall i \in [n]$
- CMax $= 2$

$\lambda_{u,j} = \lambda_{c,i} = 12\,, \forall i \in [n], j \in [m]$. Notice that the optimal solution has a cost value equal to $-2$. Second, we present the results for Instance B (Figure 4.13) in Figure 4.14. In this case, we choose


Figure 4.12: Comparison between QUBO and PUBO formulations for Extended Bin Packing on Instance A.

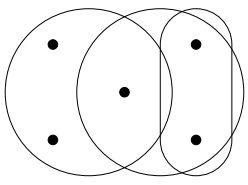

Figure 4.13: Instance B

Instance B: $n = 3$ trains (sets) and $m = 5$ groups (dots), for:

- $c_i = 1\,, \forall i \in [n]$
- $p_i = 1\,, \forall i \in [n]$
- CMax $= 2$

$\lambda_u = \lambda_c = 10$ for the QUBO formulation, respectively $\lambda_{u,j} = \lambda_{c,i} = 10\,, \forall i \in [n], j \in [m]$ for the PUBO formulation. The optimal cost value is also equal to $-2$. For both experiments, we set

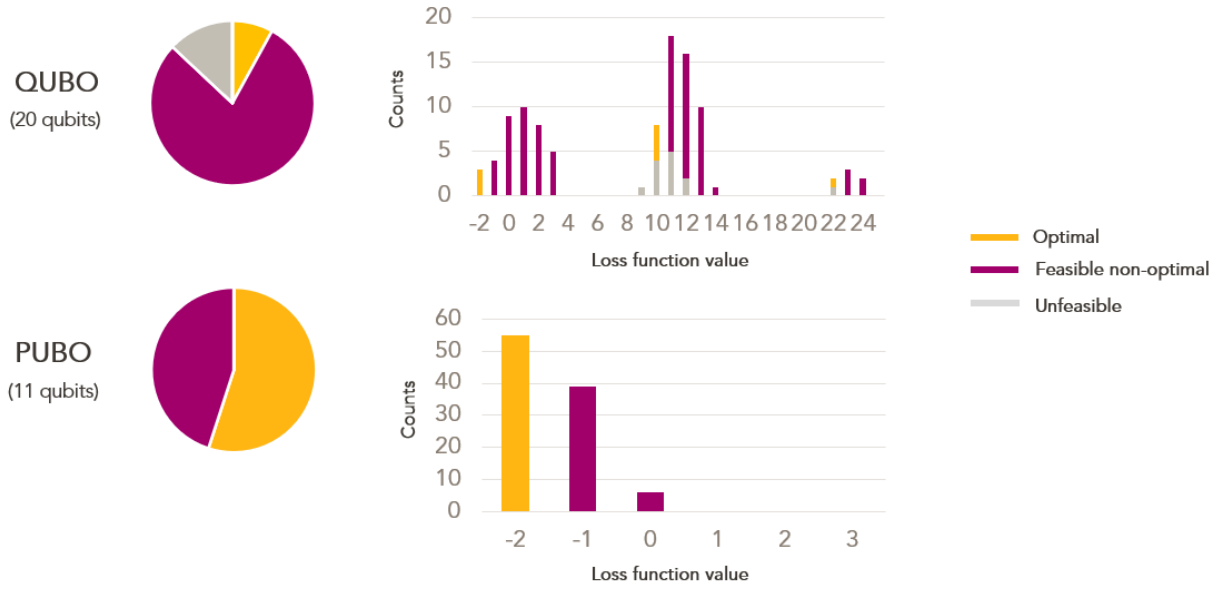


Figure 4.14: Comparison between QUBO and PUBO formulations for Extended Bin Packing on Instance B.

the depth of QAOA to 1, and make 10 shots to sample quantum states (to keep the proportion of the size of the sampling and the size of the search space reasonable). Notice that we still use COBYLA as a classical optimizer and that we implement the quantum circuit (on the 32-qubit QASM simulator) with the decomposition of Proposition 3.2.6 for the PUBO formulation.

Let us begin with an observation on the QUBO formulation. In both results on Instances A and B, we see that the optimal solution of the initial problem (in orange) can have a loss function value different from -2 and even larger than other non-optimal solutions. This is due to the introduction of additional variables $(s, r)$ in (Extended-BP-QUBO) that can take values such that the penalty terms are non-zero even if the decision variables $(x, y)$ representing a solution of the initial problem are optimal. Thus, measuring the optimal solution with loss function value 10 for Instance B is a *lucky coincidence* and has nothing to do with optimization. Thus, it points out a limit of this QUBO formulation with additional variables that *blur* the loss function. Notice that this phenomenon does not happen for the PUBO formulation (Extended-BP-PUBO) because there are no additional variables.

We discuss several observations regarding the comparison between the QUBO and the PUBO formulation from the two examples above. To solve (Extended-BP) on current quantum limited resources with the metaheuristic QAOA, the results speak in favor of the PUBO formulation. First, the number of qubits is smaller for the PUBO than for the QUBO formulation. This comes from the additional variables required to transform the constraints into quadratic penalty terms. We recall that roughly $(m + n \cdot \log_2(\text{CMax}))$ additional binary variables are used for the QUBO formulation (and none for the PUBO formulation). Second, the PUBO seems to manage more easily the constraints, leading to QAOA's output quantum state with only feasible solutions measured. We can explain it by the fact that we chose penalty functions for PUBO with binary values, 0 if the constraint is satisfied and 1 otherwise, whereas in QUBO they can take a large range of values (according to the value of CMax). This implies that the different loss function values for QUBO are gathered in many *packets* spaced by the value of penalty coefficients $\lambda_u$ and

$\lambda_c$. The larger CMax, the more possible values for additional variables, and the more packets. On the contrary, for PUBO formulation, the number of packets is restricted (because the penalty function takes only two values), thus the optimization should be easier. The results on Instances A and B attest to an easier optimization because there are no unfeasible solutions sampled by QAOA with the PUBO formulation. In other words, we only sample the *first* packet representing feasible solutions. To conclude, we observe that solving the Extended Bin Packing problem with QAOA achieves better performances with the unconstrained formulation (Extended-BP-PUBO) than with the (Extended-BP-QUBO) formulation.

## 4.4 Conclusion

In this chapter, we studied a railway timetabling problem of SNCF; we proposed two simplifications and several reformulations into unconstrained problems for each of them. We analyzed the performances of QAOA on these reformulated problems, highlighting the impact of parameters such as the depth or the shape of QAOA circuit. We also illustrated the importance of the choice of penalty functions for the second simplified problem. However, the small size of current quantum computers (around a hundred qubits), the small connectivity between qubits, and the high error rate prevent running QAOA for larger depths on larger instances. Thus, because numerical results are limited today, we will turn to more theoretical results in the next chapters to investigate the advantages of exact quantum algorithms. We will work on optimization problems that have many applications for SNCF. Precisely, we will address scheduling problems (Chapters 5 and 6), which, for instance, lie at the core of optimizing machine tasks in freight marshalling yards. We will also study robust optimization problems (Chapter 7) that are decisive in many applications. For example, robustness appears in the design of time grids that take into account uncertainties and minimize the spread of a train delay throughout the network, or in *real-time* operations where the disruptions' duration is not precisely known in advance.

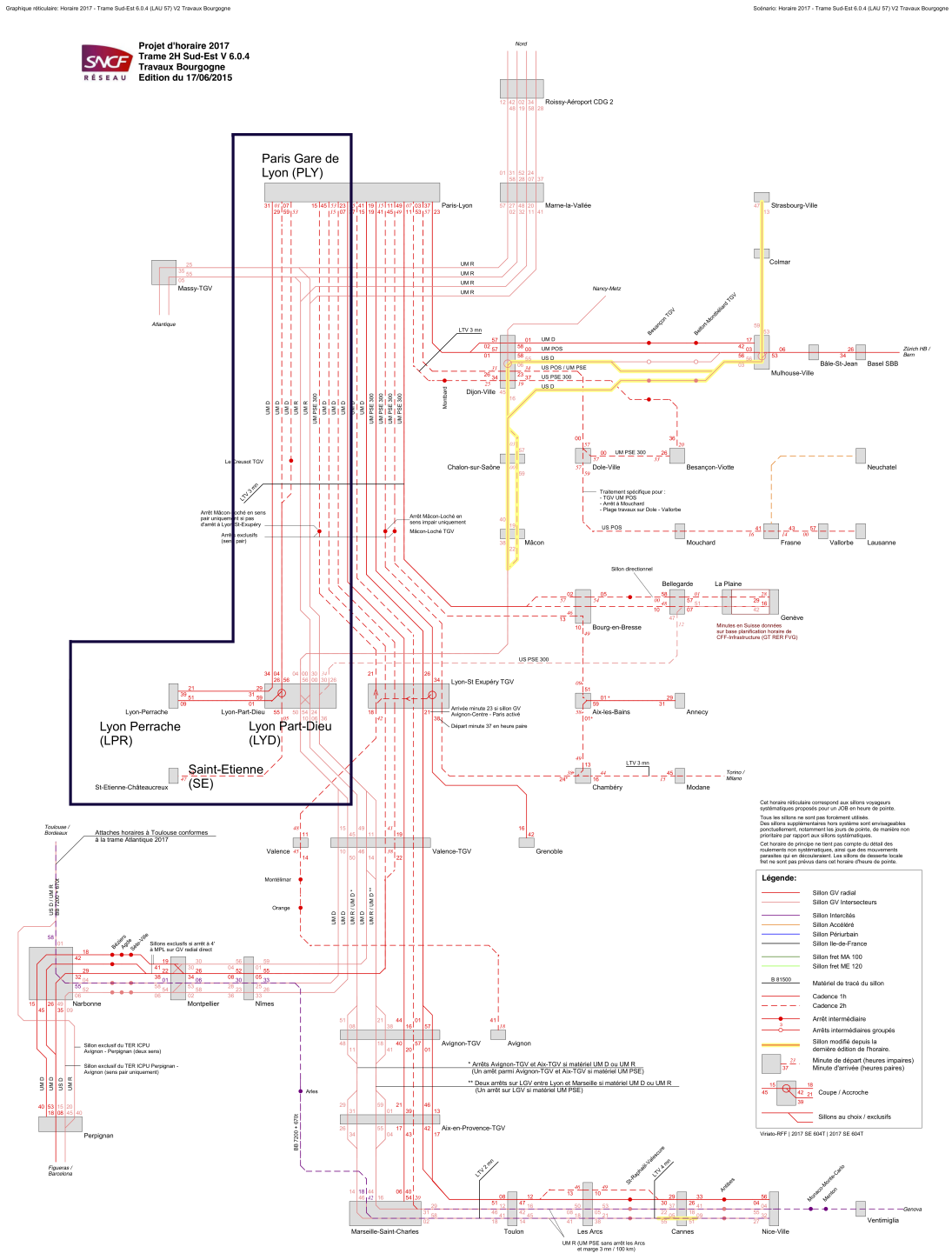


Figure 4.15: Reticular on the sector of Paris and Lyon.

# 5

# Hybrid algorithms for scheduling

The railway domain faces numerous scheduling problems. For instance, in freight marshalling yards, trains arrive at different times, and their wagons need to be separated before being reformed with wagons from other trains. Thus, operations such as uncoupling, sorting, and coupling the wagons are performed by machines that have to be scheduled. Another example is tasks scheduling for the maintenance of trains, tracks, and infrastructures, whose optimization is necessary to minimize disruptions to the service while ensuring safety and reliability.

This chapter provides a quantum-classical exact algorithm to tackle a broad class of NP-hard scheduling problems. For that, we extend well-known Dynamic Programming Across the Subsets (DPAS) recurrences. We propose a hybrid algorithm that addresses scheduling problems, relying on the new recurrences, and leads to a quantum speed-up. Our algorithm (called Q-DDPAS hereafter) is an extension of the algorithm of Ambainis et al. (2019). In particular, Q-DDPAS applies to problems with temporal constraints and non-linear objective functions. Specifically, we cover three types of problems that satisfy three different kinds of dynamic programming recurrences. Not only does Q-DDPAS apply to problems for which the dynamic programming property is based on the *addition* of optimal values of the problem on sub-instances, but it also relates to problems for which the dynamic programming naturally applies on the *composition* of optimal values of the problem on sub-instances. For each of them, the best-known classical time complexity is in $\mathcal{O}^*(2^n)$ that is reduced in $\mathcal{O}^*(pseudopol \cdot 1.728^n)$ by Q-DDPAS, where *pseudopol* is a pseudo-polynomial factor. Furthermore, we address the 3-machine flowshop problem that differs from previous problems by the nature of the recurrence property and widens the range of problems solved by the hybrid algorithm. Notice that, in this chapter, the hybrid algorithms are described as it is usually done in the algorithmic quantum literature, namely with a high-level description where quantum *boxes* interact with the classical part. We provide a rigorous and detailed description of the circuit-based implementation in Chapter 6.

The content of this chapter results from a joint work with Vincent T'Kindt (LIFAT, France) and Olivier Ploton (LIFAT, France).

## 5.1 Scheduling problems and DPAS

A scheduling problem lies in finding the optimal assignment of a set of jobs to machines over time. Each job $j$ is defined by at least a processing time $p_j$ and possibly additional data like a due date $d_j$, a deadline $\tilde{d}_j$, or even a weight $w_j$ reflecting its priority. One or more machines can process the set of jobs, however, at any time point, a machine can only process one job at a time. The computation of a schedule is done to minimize a given objective function.

In Sections 5.2 and 5.3, we consider single-machine scheduling problems. Let $[n] = \{1, \ldots, n\}$ be the set of jobs to schedule on the machine. While a solution to a single-machine scheduling problem is described by a starting time for each job on the machine, it is standard to describe instead such a solution by a permutation $\pi \in S_{[n]}$ of the $n$ jobs. Indeed, the starting times can be directly deduced from the order of jobs in the permutation and the potential constraints, thanks

to the following assumptions. First, we assume that only one job can be processed at any time on the machine. Second, we deal only with non-preemptive scheduling, meaning that a job must be run to completion once it has started. Henceforth, we use the permutation representation for the solutions. In Section 5.5, we consider the 3-machine flowshop of $n$ jobs. The definition of this problem, introduced in the above-mentioned section, makes also a solution entirely described by a permutation of $[n]$ even if there are 3 machines.

Throughout this chapter, we use the usual notation $\alpha|\beta|\gamma$, introduced by Graham et al. (1979), to describe the scheduling problem consisting of $\alpha$ machines, with the constraints $\beta$ and the criterion $\gamma$ to be minimized. For instance, $1|\tilde{d}_j|\sum_j w_j C_j$ is the problem of minimizing the total weighted completion time with deadline constraints on a single machine. More details on scheduling can be found in any textbook on the topic, e.g. the one by Pinedo (2012).

The single-machine scheduling problems addressed in this chapter are those that satisfy the Dynamic Programming Across the Subsets (DPAS) property. It means that these problems can be solved by Dynamic Programming where the optimal solution for a set of jobs $J \subseteq [n]$ is computed as the best concatenation overall $j \in J$ of the optimal solution for $J \setminus \{j\}$ and the cost of setting $j$ as the last processed job. Specifically, if we note $\mathrm{OPT}[J]$ the optimal value for processing the set of jobs $J$, the recursion is

$$\mathrm{OPT}[J] = \min_{j \in J} \mathrm{OPT}[J \setminus \{j\}] + \phi_j \left( \sum_{k \in J} p_k \right), \tag{5.1}$$

where $\phi_j$ is a function depending on job $j$. This generic recursion captures many single-machine scheduling problems as recalled in the survey of T'kindt et al. (2022), leading to the worst-case time complexity of $\mathcal{O}^*(2^n)$ to solve all these problems (where $\mathcal{O}^*$ denotes the usual asymptotic notation that ignores the polynomial factors in the complexity). Notice that DPAS is a common technique for designing exact algorithms for NP-hard problems as described by Woeginger (2003). This naturally raises the question of the existence of moderate exponential-time algorithms with a complexity $\mathcal{O}^*(c^n)$ where $c < 2$. The question has been answered positively for specific problems such as minimizing the total weighted completion time with precedence constraints in $\mathcal{O}^*((2-\epsilon)^n)$ for small $\epsilon > 0$ by Cygan et al. (2014). But, as far as we know, no generic method provides such an improvement for a broad class of scheduling problems. In this chapter, we present a hybrid algorithm that solves the problems satisfying (5.1) in $\mathcal{O}^*(1.728^n)$, sometimes with an additional pseudo-polynomial factor in the complexity that comes from the extension of the dynamic programming recurrence.

## 5.2 Additive DPAS

In this section, we present problems for which the dynamic programming recursion is based on the *addition* of optimal values of problems for sub-instances. Next, we detail the hybrid algorithm Q-DDPAS to solve these problems. But first, let us begin with an scheduling example.

### 5.2.1 A scheduling example

The NP-hard single-machine scheduling problem at hand is the minimization of the total weighted completion time with deadline constraints, often referred to as $1|\tilde{d}_j|\sum_j w_j C_j$ in the scheduling literature. The input is given, for each job $j \in [n]$, by a weight $w_j$, a processing time $p_j$ and a deadline $\tilde{d}_j$ before which the job must be completed. We define the completion time $C_j(\pi)$ of job

$j$ as the end time of the job on the machine for the permutation $\pi$. So, if $j$ starts as time $t$ for the permutation $\pi$, then $C_j(\pi) = t + p_j$. The problem aims at finding the feasible permutation for which the total weighted sum of completion times is minimal. A permutation $\pi$ is feasible if $C_j(\pi) \leq \tilde{d}_j$ for all job $j$. Thus, the problem can be formulated as follows:

$$\min_{\pi \in \Pi} \sum_{j=1}^{n} w_j C_j(\pi)\,,$$

where the set of feasible permutations is $\Pi = \{\pi \in S_{[n]} \,|\, C_j(\pi) \leq \tilde{d}_j\,, \forall j \in [n]\}$.

This problem satisfies two recurrences. For deriving them, we need to introduce the set $T := \left[\!\left[0, \sum_{j=1}^{n} p_j\right]\!\right] = \left\{0, 1, \ldots, \sum_{j=1}^{n} p_j\right\}$. For $J \subseteq [n]$ and $t \in T$, we define $\mathrm{OPT}[J,t]$ as the optimal value of the problem in which only jobs in $J$ are scheduled from time $t$. Thus, solving our nominal problem $1|\tilde{d}_j|\sum_j w_j C_j$ amounts to compute $\mathrm{OPT}[[n],0]$.

The first recurrence comes from the standard Dynamic Programming Across the Subsets (DPAS) described in (5.1). However, compared to usual DPAS, we introduce an extra parameter $t$ necessary for the solution with our hybrid algorithm as explained later. The idea of this recurrence is to get the optimal value of our problem for jobs in $J$ and starting at time $t$ by finding, over all jobs $j \in J$, the permutation that ends with $j$ with the best cost value. It is possible to do so because no matter what the optimal permutation of the first $(|J|-1)$ jobs is, the cost of setting job $j$ at the end of the permutation is always known. Indeed, the time taken to process all jobs in $J \setminus \{j\}$ is always $\sum_{k \in J \setminus \{j\}} p_k$. Thus, the completion time of $j$ is defined by $C_j = t + \sum_{k \in J} p_k$. It results that the cost of setting $j$ at the end of the permutation is $w_j(t + \sum_{k \in J} p_k)$. It also implies that the deadline constraint for job $j$ is satisfied if $t + \sum_{k \in J} p_k \leq \tilde{d}_j$. Specifically, for all $J \subseteq [n]$ and for all $t \in T$, we have

$$\mathrm{OPT}[J,t] = \min_{j \in J} \mathrm{OPT}[J \setminus \{j\}, t] + \begin{cases} w_j\left(t + \sum_{k \in J} p_k\right) & \text{if } t + \sum_{k \in J} p_k \leq \tilde{d}_j \\ +\infty & \text{otherwise} \end{cases} \tag{5.2}$$

initialized by $\mathrm{OPT}[\emptyset] = 0$.

The second recurrence generalizes the previous one. For this recurrence, the principle of computing $\mathrm{OPT}[J,t]$ is similar to (5.2), but instead of setting one job at the end of the permutation, we choose $|J|/2$ jobs and set them to be the half last jobs of the permutation. Specifically, for all $J \subseteq [n]$ of even cardinality and $t \in T$, we have

$$\mathrm{OPT}[J,t] = \min_{\substack{X \subseteq J \\ |X| = |J|/2}} \left\{\mathrm{OPT}[X,t] + \mathrm{OPT}[J \setminus X, t + \sum_{i \in X} p_i]\right\}, \tag{5.3}$$

initialized by, $\forall j \in [n]$ and $t \in T$, $\mathrm{OPT}[\{j\},t] = \begin{cases} w_j(p_j + t) & \text{if } \tilde{d}_j \geq p_j + t \\ +\infty & \text{otherwise} \end{cases}$.

For a given $X \subseteq J$ of size $|J|/2$, recurrence (5.3) computes the best permutation of jobs in $X$ starting at time $t$, and the best permutation of jobs in $J \setminus X$ starting at time $t + \sum_{k \in X} p_k$ as we know that, as before, no matter what is the optimal permutation for jobs in $X$, the time taken to process them all is exactly $\sum_{k \in X} p_k$.

The two above recurrences have been illustrated with problem $1|\tilde{d}_j|\sum_j w_j C_j$. In the next section, we propose a general formulation of these recurrences that will be used to elaborate our algorithm Q-DDPAS as general as possible to solve a broad class of scheduling problems.

### 5.2.2 General formulation of recurrences

Let us consider the following general scheduling problem:

$$\mathcal{P}: \qquad \min_{\pi \in \Pi} f(\pi)\,,$$

where $\Pi \subseteq S_{[n]}$ is the set of feasible permutations of $[n]$ according to given constraints and $f$ is the objective function. We introduce a related problem $P$ useful for deriving the dynamic programming recursion, for which we specify the instance: for $J \subseteq [n]$ and $t \in \mathbb{Z}$, we define

$$P(J,t): \qquad \min_{\pi \in \Pi(J,t)} f(\pi, J, t) \tag{5.4}$$

as the nominal scheduling problem $\mathcal{P}$ that schedules only jobs in $J$ and starts the schedule at time $t$. Let us note $\mathrm{OPT}[J,t]$ the optimal value of $P(J,t)$. It results that solving $\mathcal{P}$ amounts to solving $P([n],0)$, and it can be performed by our algorithm Q-DDPAS if the related problem $P$ satisfies the two recurrences (Add-DPAS) and (Add-D-DPAS) below. Henceforth, we denote by $2^{[n]}$ the set of all subsets of $[n]$. Let us introduce the first recurrence.

**Property 5.2.1** (Additive DPAS)**.** *There exists a function $g : 2^{[n]} \times [n] \times T \to \mathbb{R}$, computable in polynomial time, such that, for all $J \subseteq [n]$ and for all $t_0 \in T$,*

$$\mathrm{OPT}[J,t_0] = \min_{j \in J} \Big\{ \mathrm{OPT}[J \setminus \{j\}, t_0] + g(J,j,t_0) \Big\} \tag{Add-DPAS}$$

*initialized by* $\mathrm{OPT}[\emptyset, t_0] = 0$.

**Lemma 5.2.2.** *Dynamic programming* (Add-DPAS) *solves $\mathcal{P}$ in $\mathcal{O}^*(2^n)$.*

*Proof.* We solve Equation (Add-DPAS) for all $J$ such that $|J| = k$, and for $t_0 = 0$, starting from $k = 1$ to $k = n$. For a given $J$, the values $\{\mathrm{OPT}[J \setminus \{j\}, 0] : j \in J\}$ are known, so $\mathrm{OPT}[J,0]$ is computed in time $poly(n) \cdot k$ according to Equation (Add-DPAS) (the computation of $g$ is polynomial). Eventually, the total complexity of computing $\mathrm{OPT}[[n],0]$ is

$$\sum_{k=1}^{n} poly(n) k \binom{n}{k} = poly(n) \cdot n \cdot 2^{n-1} = \mathcal{O}^*(2^n)\,.$$

□

Hereafter, we commit a slight abuse of language by letting (Add-DPAS) both refer to the property satisfied by a given optimization problem and to the resulting dynamic programming algorithm. Notice the presence of the additional parameter $t_0$ in the above definition, which is typically absent in the scheduling literature. In particular, $t_0$ is a constant throughout the whole recursion (Add-DPAS) and does not impact the resulting computational complexity. The use of that extra parameter in $T$ shall be necessary later when applying our hybrid algorithm.

Property 5.2.1 expresses that finding the optimal value of $P$ for jobs in $J$ and starting at time $t$ is done by finding over all jobs $j \in J$ the permutation that ends by $j$ with the best cost value. Function $g$ represents the cost of $j$ being the last job of the permutation. Notice that isolating the last job of the permutation is a usual technique in scheduling as displayed in (5.1). In the second recurrence below, we provide a similar scheme, where instead of one job, we *isolate* half of the jobs in $J$, turning the computation of $g$ to the solution of another problem on a sub-instance with $|J|/2$ jobs.

**Property 5.2.3** (Additive Dichotomic DPAS)**.** *There exist two functions* $\tau_{shift} : 2^{[n]} \times 2^{[n]} \times T \to T$ *and* $h : 2^{[n]} \times 2^{[n]} \times T \to \mathbb{R}$*, computable in polynomial time, such that, for all* $J \subseteq [n]$ *of even cardinality, and for all* $t \in T$*,*

$$\mathrm{OPT}[J,t] = \min_{\substack{X \subseteq J \\ |X|=|J|/2}} \Big\{ \mathrm{OPT}[X,t] + h(J,X,t) + \mathrm{OPT}[J \setminus X, \tau_{shift}(J,X,t)] \Big\} \qquad \text{(Add-D-DPAS)}$$

*initialized by the values* $\mathrm{OPT}[\{j\},t]$ *for each* $j \in [n]$ *and* $t \in T$*.*

For a given $X \subseteq J$, the above recursion computes the best permutation of jobs in $X$ starting at time $t$, and the best permutation of jobs in $J \setminus X$ starting at time $\tau_{\text{shift}}$, adding the function $h$ that represents the cost of the concatenation between these two permutations.

**Remark 5.2.4.** *We observe that problem* (5.4) *satisfies recurrence* (Add-DPAS) *if and only if it satisfies* (Add-D-DPAS)*. This can be seen by developing recursively both recurrences, which essentially leads to optimization problems over* $\pi \in S_{[n]}$*, whose objective functions respectively involve* $g$ *in the first case and* $h$ *and* $\tau_{shift}$ *in the second case. Here, one readily verifies that* $g$ *can then be defined from* $h$ *and* $\tau_{shift}$ *and reciprocally.*

Despite the previous remark, the two recurrences differ on the size of the subsets considered along the recursions, leading to different formulations and therefore require more or less subproblems to be solved at optimal in the dynamic programming process. This is formalized in the following proposition.

Note that we use the notation $f_1(n) = \omega(f_2(n))$ if $f_1$ dominates asymptotically $f_2$.

**Lemma 5.2.5.** *Dynamic programming* (Add-D-DPAS) *solves* $\mathcal{P}$ *in* $\omega(|T| \cdot 2^n)$*.*

*Proof.* First, we note that to solve $\mathcal{P}$ with (Add-D-DPAS), $n$ must be a power of 2. If this is not the case, we can always transform the instance such that we fall back into the previous case. Thus, without loss of generality, we suppose that $n = 2^N$ for $N \in \mathbb{N}$. We solve Equation (Add-D-DPAS) for all $J$ such that $|J| = 2^k$, and for all $t \in T$, starting from $k = 1$ to $k = N$. For a given $J$, the values $\{\mathrm{OPT}[X,t'] : X \subseteq J \text{ s.t. } |X| = |J|/2\,, t' \in T\}$ are known, so $\mathrm{OPT}[J,t]$ is computed in time $poly(n)\binom{2^k}{2^{k-1}}$ according to Equation (Add-D-DPAS) (the computation of $\tau_{\text{shift}}$ and $h$ is polynomial). Thus, computing all $\mathrm{OPT}[J,t]$ for any $J$ of size $2^k$ and $t \in T$ is done in time $|T|poly(n)\binom{2^k}{2^{k-1}}\binom{n}{2^k}$. Eventually, the total complexity is equal to

$$C(n) = |T|poly(n) \sum_{k=1}^{N} \binom{2^k}{2^{k-1}} \binom{n}{2^k}.$$

Second, we compute the complexity of (Add-D-DPAS). For that, we consider the sequence $(C(2^i))_{i \in \mathbb{N}}$, knowing that for families of instances with a size different from a power of 2, we transform them artificially into families of instances of size of the following power of 2. Let $n = 2^i$ for $i \in \mathbb{N}$. A lower bound on $C(n)$ is the sum of the two last terms:

$$C(n) > |T|poly(n) \left( \binom{n}{n/2} + \binom{n}{n/2} \binom{n/2}{n/4} \right) \approx A|T|poly(n) \frac{2^{1.5n}}{n},$$

where $A$ is a constant. The asymptotic equivalent is readily obtained with the Stirling equivalent for factorials, $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, for $n \in \mathbb{N}$. Thus, $C$ dominates asymptotically $n \mapsto |T| \cdot 2^n$. In other words, $C(n) = \omega(|T| \cdot 2^n)$. □

The previous lemma leads to the conclusion that solving $\mathcal{P}$ with (Add-DPAS) is faster than with (Add-D-DPAS). However, in the next section, we describe a hybrid algorithm Q-DDPAS that improves the complexity of solving $\mathcal{P}$ by combining recurrences (Add-DPAS) and (Add-D-DPAS) with a quantum subroutine.

### 5.2.3 Hybrid algorithm for Additive DPAS

In this subsection, we describe our hybrid algorithm Q-DDPAS adapted from the work of Ambainis et al. (2019). This latter combines Quantum Minimum Finding, proposed by Dürr and Høyer (1996) to find the minimum of an unsorted table, and dynamic programming. The algorithm of Ambainis et al. (2019) applies to NP-hard vertex ordering problems, such as the Traveling Salesman Problem (TSP) or the Minimum Set Cover problem. The problems of interest must satisfy a specific property which implies that they can be solved by classical dynamic programming in $\mathcal{O}^*(c^n)$, where $c$ is usually not smaller than 2. The hybrid algorithm of Ambainis et al. (2019) reduces the complexity to $\mathcal{O}^*(c^n_{\text{quant}})$ for $c_{\text{quant}} < c$. As an example, Held and Karp (1970) dynamic programming solves the TSP in $\mathcal{O}^*(2^n)$ whereas the hybrid algorithm achieves to solve it in $\mathcal{O}^*(1.728^n)$ by combining the dynamic programming recurrence of Held and Karp with Quantum Minimum Finding. Following this work, other NP-hard problems have been tackled with the idea of combining Grover Search (or Quantum Minimum Finding) and classical dynamic programming. For instance, this has led to quantum speed-ups for the Steiner Tree problem (Miyamoto et al., 2020) and the graph coloring problem (Shimizu and Mori, 2022).

Henceforth, we propose to widen the range of problems tackled by the algorithm of Ambainis et al. (2019) to NP-hard scheduling problems, essentially by introducing a temporal dimension to the dynamic programming recursion. Notice that the hybrid algorithm assumes to have a quantum random access memory (QRAM) (Giovannetti et al., 2008), namely, to have a classical data structure that stores classical information but can answer queries in quantum superposition. We underline that this latter assumption is strong because QRAM is not yet available on current universal quantum hardware. First, let us introduce the Quantum Minimum Finding algorithm of Dürr and Høyer (1996), which constitutes a fundamental subroutine in our algorithm. This algorithm essentially applies several times Grover Search (Grover, 1996) to provide a quadratic speedup for the search of a minimum element in an unsorted table.

**Definition 5.2.6** (Quantum Minimum Finding (Dürr and Høyer, 1996))**.** *Let $f : [n] \to \mathbb{Z}$ be a function. Quantum Minimum Finding computes the minimum value of $f$ and the corresponding minimizer* $\arg\min_{i\in[n]}\{f(i)\}$*. The complexity of Quantum Minimum Finding is $\mathcal{O}\left(\sqrt{n}\cdot C_f(n)\right)$, where $\mathcal{O}(C_f(n))$ is the complexity of computing a value of $f$.*

**Remark 5.2.7** (Success probability and bounded-error algorithm (Bernstein and Vazirani, 1993))**.** *Dürr and Høyer (1996) prove that Quantum Minimum Finding computes the minimum value with a probability of success strictly larger than $\frac{1}{2}$, independent of $n$. Thus, for $\epsilon > 0$, finding the minimum value with probability $(1-\epsilon)$ is achieved by repeating $\mathcal{O}(\log\frac{1}{\epsilon})$ times Quantum Minimum Finding. Henceforth, we refer to this statement when we write that Quantum Minimum Finding finds the minimum value* with high probability*. Equivalently, we say that this is a* bounded-error *algorithm. More generally, in the rest of the chapter, we call a* bounded-error algorithm *an algorithm that provides the optimal solution with a probability as close to 1 as we want by repeating it a number of times independent of the instance size.*

Next, we describe the algorithm of Ambainis et al. (2019) adapted for our Additive DPAS recurrences which implies extra parameters in $T$. We call it Q-DDPAS, which consists essentially

of calling recursively twice Quantum Minimum Finding and computing classically the left terms. Without loss of generality, we assume that 4 divides $n$. This can be achieved by adding at most three fake jobs and, therefore, does not change the algorithm complexity. Q-DDPAS consists of two steps. First, we compute classically by (Add-DPAS) the optimal values of $P$ on sub-instances of $n/4$ jobs and for all starting times $t \in T$. Second, we call recursively two times Quantum Minimum Finding with (Add-D-DPAS) to find optimal values of $P$ on sub-instances of $n/2$ jobs starting at any time $t \in T$, and eventually of $n$ jobs starting at $t = 0$ (corresponding to the optimal value of the nominal problem $\mathcal{P}$). Specifically, we describe Q-DDPAS in Algorithm 2.

**Algorithm 2:** Q-DDPAS for Additive DPAS

**Input:** Problem $P$ satisfying (Add-DPAS) and (Add-D-DPAS)
**Output:** $\mathrm{OPT}[[n], 0]$ with high probability
**begin classical part**
  **for** $X \subseteq [n]$ *such that* $|X| = n/4$, *and* $t \in T$ **do**
1  Compute $\mathrm{OPT}[X, t]$ with (Add-DPAS) and store the results in the QRAM;
**begin quantum part**
2  Apply Quantum Minimum Finding with (Add-D-DPAS) to find $\mathrm{OPT}[[n], 0]$;
3  To get values for the Quantum Minimum Finding above (the values $\mathrm{OPT}[J, t]$ for $J \subseteq [n]$ of size $n/2$ and $t \in T$), apply Quantum Minimum Finding with (Add-D-DPAS);
4  To get values for the Quantum Minimum Finding above (the values $\mathrm{OPT}[X, t']$ for $X \subseteq [n]$ of size $n/4$ and $t' \in T$), get them on the QRAM

**Theorem 5.2.8.** *The bounded-error algorithm Q-DDPAS (Algorithm 2) solves $\mathcal{P}$ in $\mathcal{O}^*(|T| \cdot 1.754^n)$.*

The detailed proof of the correctness of the algorithm, involving the description of the gate implementation, is detailed in Chapter 6, with all the low-level details for implementing the algorithm. Next, we provide a high-level proof, but before, we introduce some upper bounds necessary to derive the complexities in the proof.

**Observation 5.2.9.** *We define the binary entropy of $\epsilon \in ]0, 1[$ by*

$$H(\epsilon) = -(\epsilon \log_2(\epsilon) + (1 - \epsilon) \log_2(1 - \epsilon)) .$$

*We remind some useful upper bounds of binomial coefficients (Ambainis et al., 2019):*

$$\binom{n}{k} \leq 2^{H(\frac{k}{n}) \cdot n}, \ \forall k \in [\![1, n]\!] \qquad \textit{and} \qquad \sum_{i=1}^{k} \binom{n}{i} \leq 2^{H(\frac{k}{n}) \cdot n}, \forall k \in \left[\!\left[1, \frac{n}{2}\right]\!\right] .$$

*Thus, this leads to the following upper bounds to compute the complexities of interest:*

$$\sum_{i=k}^{n/4} \binom{n}{k} \leq 2^{0.811n} , \quad \sum_{k=1}^{0.945 \cdot n/4} \binom{n}{k} \leq 2^{0.789n} , \quad \sqrt{\binom{n}{n/2}\binom{n/2}{n/4}} \leq 2^{0.75n} ,$$

*and*

$$\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}\binom{n/4}{0.945\cdot n/4}} \leq 2^{0.789n}\,.$$

*Proof.* Hereafter, we provide a high-level proof of Theorem 5.2.8. Let us compute the complexity of each part of Q-DDPAS algorithm.

- Classical part: computing all $\mathrm{OPT}[X,t]$ for all $X$ of size $n/4$ and for all $t \in T$ (Step 1) is done by (Add-D-DPAS) in time $\mathcal{O}^*\left(|T|\cdot\sum_{k=1}^{n/4} k\binom{n}{k}\right) = \mathcal{O}^*(|T|\cdot 2^{0.811n})$.
- Quantum part: according to Quantum Minimum Finding complexity (Definition 5.2.6), computing $\mathrm{OPT}[[n],0]$ with Quantum Minimum Finding (Step 2) is done in $\mathcal{O}\left(\sqrt{\binom{n}{n/2}}\cdot C_1(n)\right)$, where $C_1(n)$ is the complexity of computing $\mathrm{OPT}[J,t]$ for a $J$ of size $n/2$ and $t \in T$. *The essence of the quantum advantage here is that we do not need to enumerate all sets $J$ and all time $t$ but we apply the Quantum Minimum Finding in parallel to all at once.* Notice that $\binom{n}{n/2}$ is the number of balanced bi-partitions of $[n]$, namely the number of elements we search over to find the minimum of Equation (Add-D-DPAS) when computing $\mathrm{OPT}[[n],0]$. Thus, $C_1(n)$ is exactly the complexity of Quantum Minimum Finding applied on Step 3, namely $C_1(n) = \mathcal{O}\left(\sqrt{\binom{n/2}{n/4}}\cdot C_2(n)\right)$ where $C_2(n)$ is the complexity of computing $\mathrm{OPT}[X,t']$ for $X$ of size $n/4$ and $t' \in T$. Those values are already computed and stored in the QRAM (Step 1), namely, $C_2(n) = \mathcal{O}^*(1)$. Thus, the quantum part complexity is $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}}\right) = \mathcal{O}^*(2^{0.75n})$.

Eventually, Q-DDPAS complexity is the maximum of the classical and the quantum part complexity. Specifically, the total complexity is $\mathcal{O}^*(|T|\cdot 2^{0.811n}) = \mathcal{O}^*(|T|\cdot 1.754^n)$. □

We observe that the complexity of Q-DDPAS can be further reduced by performing a third call to Equation (Add-D-DPAS) as suggested by Ambainis et al. (2019).

**Observation 5.2.10.** *A slight modification of Q-DDPAS reduces the complexity to* $\mathcal{O}^*(|T|\cdot 1.728^n)$.

*Proof.* The slight modification of Q-DDPAS amounts to adding a level of recurrence in the quantum part so that the complexity of the classical part reduces whereas the complexity of the quantum part increases so that both are equal and thus minimize the total complexity. The third call searches for the best concatenation among all the bi-partitions of size $(0.945\cdot\frac{n}{4}, 0.055\cdot\frac{n}{4})$ (that are integers asymptotically), i.e. solving

$$\mathrm{OPT}[J,t] = \min_{\substack{X\subseteq J\\ |X|=0.945|J|}} \left\{\mathrm{OPT}[X,t] + h(J,X,t) + \mathrm{OPT}[J\setminus X, \tau_{\mathrm{shift}}(J,X,t)]\right\}.$$

The classical part computes all $\mathrm{OPT}[X,t]$ for $X$ of size $0.945\cdot\frac{n}{4}$ and $0.055\cdot\frac{n}{4}$, in $\mathcal{O}^*(1.728^n)$. The quantum part applies three levels of recurrence of Quantum Minimum Finding, computing the minimum over functions with a domain of size $\binom{n}{n/2}$, $\binom{n/2}{n/4}$ and $\binom{n/4}{0.945\cdot n/4}$ respectively. Its complexity is then $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}\binom{n/4}{0.945\cdot n/4}}\right) = \mathcal{O}^*(1.728^n)$ (see Observation 5.2.9). □

Notice that the classical part of Q-DDPAS can be replaced by any classical algorithm $\mathcal{A}$, if $\mathcal{A}$ computes in $\mathcal{O}^*(|T|\cdot 1.728^n)$ all $\mathrm{OPT}[X,t]$ for $X \subseteq [n]$ of size $n/4$ and $t \in T$. Moreover, if

$\mathcal{A}$ happens to reduce the classical part complexity $\mathcal{O}^*(|T| \cdot c^n)$ for $c < 1.728$, the complexity of Q-DDPAS can also be reduced in the same spirit as the slight modification of Observation 5.2.10.

The application of Q-DDPAS for Additive DPAS to the specific problem $1|\tilde{d}_j|\sum_j w_j C_j$ introduced in Section 5.2.1 is given in Section 5.4.1, together with other scheduling examples. Before introducing other types of problems tackled by Q-DDPAS in the next section, we provide some insights to underline why the use of the quantum subroutine Quantum Minimum Finding in Q-DDPAS must be carefully combined with classical computation to achieve a quantum speed-up.

**Remark 5.2.11.** *Solving $\mathcal{P}$ with* (Add-DPAS) *and replacing each classical computation of the minimum by the quantum subroutine Quantum Minimum Finding would not improve the best classical complexity. Indeed, the complexity would be*

$$\sum_{k=1}^{n} poly(n)\sqrt{k}\binom{n}{k} = \mathcal{O}^*(2^n)\,.$$

**Remark 5.2.12.** *Solving $\mathcal{P}$ exclusively by recursive calls to Quantum Minimum Finding (thus avoiding the classical computations for sets of size $n/4$) would not improve the classical complexity. Using recurrence* (Add-D-DPAS)*, which is the quantum part of Algorithm 2 with roughly $\log_2(n)$ recursive calls, would give a complexity in*

$$\mathcal{O}\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}\cdots\binom{2}{1}}\right)$$

*that is worse than $\mathcal{O}^*(2^n)$. Using recurrence* (Add-DPAS) *would be even worse because it would require $n$ recursive calls leading to the complexity*

$$\mathcal{O}(\sqrt{n(n-1)\dots 1})\,.$$

## 5.3 Composed DPAS

In this section, we study scheduling problems whose constraints enable only the *composition* of problems on sub-instances. We describe the adaptation of Q-DDPAS for these problems.

### 5.3.1 A scheduling example

We begin with the specific problem of minimizing the total weighted number of late jobs with release date constraints, often referred to as $1|r_j|\sum w_j U_j$ in the literature. The input is given by, for each job $j \in [n]$, a weight $w_j$, a processing time $p_j$, a release date $r_j$ that is the time from which the job can be scheduled (and not before), and a due date $d_j$ that indicates the time after which the job is late. Thus, a job $j$ is late in permutation $\pi$ if its completion time is larger than $d_j$, namely if $C_j(\pi) > d_j$. We name $U_j(\pi) = \mathbb{1}_{C_j(\pi)>d_j}$ its indicator function of lateness. This problem aims at finding the feasible permutation, namely where each job starts after its release date, for which the total weighted number of late jobs is minimal. Thus, the problem can be formulated as follows:

$$\min_{\pi\in\Pi} \sum_{j=1}^{n} w_j U_j(\pi)\,,$$

where the set of feasible solutions is $\Pi = \{\pi \in S_{[n]} \,|\, C_j(\pi) \geq r_j + p_j\}$.

This problem does not satisfy the recurrences (Add-DPAS) and (Add-D-DPAS) because the release date constraints do not allow the addition of sub-instances. Let us take the example of (Add-D-DPAS). The starting time of the second half of jobs $J \setminus X$ in (Add-D-DPAS) can be known only if we know the optimal permutation of the first half of jobs, which is in opposition with the dynamic programming principle. Indeed, the release dates enable empty slots in the scheduling on the first half of jobs such that the time to process them all is not always equal to $\sum_{k\in X} p_k$ and can be larger.

This observation leads to different recurrences, where the time to process the jobs would be known by dynamic programming. For that, we define an auxiliary problem on which the recurrences apply and we introduce a new set of parameters $E := \left[\!\left[0, \sum_{j=1}^{n} w_j\right]\!\right]$. For $J \subseteq [n]$, $t \in T := \left[\!\left[0, \sum_{j=1}^{n} p_j\right]\!\right] \cup \{+\infty\}$ and $\epsilon \in E$, we note $\mathrm{OPT}[J, t, \epsilon]$ the minimum makespan, i.e. the completion time of the last job, for jobs in $J$ beginning at time $t$ where the weighted number of late jobs is exactly $\epsilon$. Notice that by convention, $\mathrm{OPT}[J, t, \epsilon] = +\infty$ if there is no feasible solution, i.e. if $\left\{\pi \in S_J : C_j(\pi) \geq \max(t, r_j) + p_j \,, \forall j \in J \text{ and } \sum_{j\in J} w_j U_j(\pi) = \epsilon\right\} = \emptyset$. Thus, our initial problem $1|r_j|\sum w_j U_j$ is

$$\min_{\epsilon\in E}\{\epsilon : \mathrm{OPT}[[n], 0, \epsilon] < +\infty\}\,.$$

The following recurrence that satisfies the auxiliary problem is inspired by the work of Lawler (1990) for the problem of minimizing the total weighted number of late jobs on a single machine under preemption and release date constraints ($1|r_j, pmtn|\sum w_j U_j$). For $J \subseteq [n]$, $t \in T$, $\epsilon \in E$,

$$\mathrm{OPT}[J, t, \epsilon] = \min_{j\in J}\Big\{\underbrace{\mathrm{OPT}\Big[\{j\}, \mathrm{OPT}[J \setminus \{j\}, t, \epsilon], 0\Big]}_{\text{job } j \text{ is not late}}, \underbrace{\mathrm{OPT}\Big[\{j\}, \mathrm{OPT}[J \setminus \{j\}, t, \epsilon - w_j], w_j\Big]}_{\text{job } j \text{ is late}}\Big\}\,.$$

In this recurrence, for each $j \in J$, we impose $j$ as the last job of the permutation and distinguish two cases, whether it is late or not. Notice that the starting time of $j$ is known and equal to $\mathrm{OPT}[J \setminus \{j\}, t, .]$ which represents the value for the time parameter. For $j \in [n]$, $t \in T$ and $\epsilon \in E$, the recurrence is initialized by

$$\mathrm{OPT}[\{j\}, t, \epsilon] = \begin{cases} C_j := \max(t, r_j) + p_j, & \text{if } C_j \leq d_j \text{ and } \epsilon = 0 \\ +\infty, & \text{if } C_j > d_j \text{ and } \epsilon = 0\,, \text{ or if } C_j \leq d_j \text{ and } \epsilon = w_j \\ C_j, & \text{if } C_j > d_j \text{ and } \epsilon = w_j \\ +\infty, & \text{if } \epsilon \in [\![1, w_j - 1]\!] \cup [\![w_j + 1, \sum_{k=1}^{n} w_k]\!] \end{cases}$$

This recurrence generalizes into the following *dichotomic* version for which, instead of setting the last job of the permutation, we set the half last jobs. For all $J \subseteq [n]$ of even cardinality, $t \in T$ and $\epsilon \in E$,

$$\mathrm{OPT}[J, t, \epsilon] = \min_{\substack{\epsilon'\in E \\ X\in J : |X| = |J|/2}} \Big\{\mathrm{OPT}\Big[X, \mathrm{OPT}[J \setminus X, t, \epsilon - \epsilon'], \epsilon'\Big]\Big\}\,,$$

initialized by the same values of $\mathrm{OPT}[\{j\}, t, \epsilon]$ for $j \in [n]$, $t \in T$ and $\epsilon \in E$. Next, we provide generic recurrences to consider problems for which the composition of sub-instances is possible.

### 5.3.2 General formulation of recurrence

Let us consider a scheduling problem with $n$ jobs

$$\mathcal{P}: \qquad \min_{\pi\in\Pi} f(\pi)\,,$$

where $\Pi \subseteq S_{[n]}$ is the set of feasible permutations of $[n]$ according to given constraints and $f$ is the objective function. Following the example detailed previously, we consider an auxiliary problem $\mathcal{P}'$ useful for deriving the dynamic programming recursion, for which we specify the instance: for $J \subseteq [n]$ the jobs to be scheduled, $t \in \mathbb{Z}$ the starting time of the schedule and $\epsilon \in \mathbb{Z}$, we define

$$P'(J,t,\epsilon): \qquad \min_{\pi\in\Pi'(J,t,\epsilon)} f'(\pi,J,t,\epsilon)\,, \tag{5.5}$$

where $f'$, respectively $\Pi'$, is the objective function, respectively the feasible set, and are different from those of $\mathcal{P}$. We assume that solving $\mathcal{P}$ amounts to finding the smallest $\epsilon \in \mathbb{Z}$ such that the auxiliary problem $\mathcal{P}'$ is bounded. Specifically,

$$\mathcal{P}: \quad \min_{\epsilon\in\mathbb{Z}}\Big\{\epsilon : \mathrm{OPT}[[n],0,\epsilon] < +\infty\Big\}\,. \tag{5.6}$$

To solve the nominal problem $\mathcal{P}$ by classical dynamic programming, problem $\mathcal{P}'$ must satisfy recurrence (Comp-DPAS) or recurrence (Comp-D-DPAS) below (as in Remark 5.2.4, we can state that a problem satisfies one if and only if it satisfies the other one). As we explain later, solving $\mathcal{P}$ with our hybrid algorithm requires problem $P'$ to satisfy the two recurrences.

**Property 5.3.1** (Composed DPAS)**.** *For all $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$,*

$$\mathrm{OPT}[J,t,\epsilon] = \min_{\substack{\epsilon'\in E\\ j\in J}}\Big\{\mathrm{OPT}\Big[\{j\},\mathrm{OPT}[J\setminus\{j\},t,\epsilon-\epsilon'],\epsilon'\Big]\Big\}\,, \tag{Comp-DPAS}$$

*initialized by the values of* $\mathrm{OPT}[\{j\},t,\epsilon]$ *for all $j \in [n]$, $\epsilon \in E$ and $t \in T$. Notice that for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$, we adopt the convention* $\mathrm{OPT}[J,t,\epsilon] = +\infty$ *for $\epsilon \notin E$.*

Recurrence (Comp-DPAS) differs from recurrence (Add-DPAS) in two aspects. First, the optimal values of the problem on sub-instances are composed, and not added, because of the nature of the constraints. Second, the search for the minimum value is done not only over all jobs in $J$, but also over all values in $E$. More precisely, for a given $\epsilon_0 \in E$, the optimal value of $P'(J,t,\epsilon_0)$ is the minimum value of all possible composition of optimal values of the problem on sub-instances with parameters $\epsilon_1$ and $\epsilon_2$ such that $\epsilon_1 + \epsilon_2 = \epsilon_0$. We have the following result.

**Lemma 5.3.2.** (Comp-DPAS) *solves $\mathcal{P}$ in $\mathcal{O}^*(|E|^3\cdot|T|\cdot 2^n)$.*

*Proof.* Let $\epsilon_0 \in E$. Similarly to the proof of Lemma 5.2.2, we show that (Comp-DPAS) solves $P'([n],0,\epsilon_0)$ in $\mathcal{O}^*(|E|^2\cdot|T|\cdot 2^n)$. Indeed, to compute $\mathrm{OPT}[[n],0,\epsilon_0]$, we need to solve Equation (Comp-DPAS) for all $J$ such that $|J| = k$ starting from $k = 1$ to $k = n$, and for all $t \in T$ and $\epsilon \in E$. For a given $J$, $t \in T$ and $\epsilon \in E$, the values $\{\mathrm{OPT}[J\setminus\{j\},t',\epsilon'] : j \in J, t' \in T, \epsilon' \in E\}$ and $\{\mathrm{OPT}[\{j\},t',\epsilon'] : j \in J, t' \in T, \epsilon' \in E\}$ are known, so $\mathrm{OPT}[J,t,\epsilon]$ is computed in time $|E|\cdot k$ according to Equation (Comp-DPAS). Eventually, the total complexity of computing $\mathrm{OPT}[[n],0,\epsilon_0]$ is

$$\sum_{k=1}^{n}|T|\cdot|E|^2\cdot k\binom{n}{k} = \mathcal{O}^*(|T|\cdot|E|^2\cdot 2^n)\,.$$

Moreover, solving $\mathcal{P}$ amounts to solving $P'([n], 0, \epsilon)$, for all $\epsilon \in E$, according to (5.6). The complexity results directly from the above complexity of computing $\mathrm{OPT}[[n], 0, \epsilon_0]$, for $\epsilon_0 \in E$. □

The auxiliary problem $P'$ must satisfy the following recurrence (Comp-D-DPAS) in addition to recurrence (Comp-DPAS).

**Property 5.3.3** (Composed Dichotomic DPAS)**.** *For all $J \subseteq [n]$ of even cardinality, $t \in T$ and $\epsilon \in E$,*

$$\mathrm{OPT}[J, t, \epsilon] = \min_{\substack{\epsilon' \in E \\ X \in J : |X| = |J|/2}} \left\{ \mathrm{OPT}\Big[X, \mathrm{OPT}[J \setminus X, t, \epsilon - \epsilon'], \epsilon'\Big] \right\}, \qquad \text{(Comp-D-DPAS)}$$

*initialized by the values of $\mathrm{OPT}[\{j\}, t, \epsilon]$ for all $j \in [n]$, $t \in T$ and $\epsilon \in E$.*

**Lemma 5.3.4.** (Comp-D-DPAS) *solves $\mathcal{P}$ in $\omega(|E|^3 \cdot |T| \cdot 2^n)$.*

*Proof.* This proof is essentially the same as the one of Lemma 5.2.5 with the same modifications that for the proof of Lemma 5.3.2. □

As for the Additive DPAS, we notice that, with a classical dynamic programming algorithm, the time complexity to solve $\mathcal{P}$ with recurrence (Comp-DPAS) is better than with recurrence (Comp-D-DPAS). Next, we show that the hybrid algorithm applied to problems satisfying Additive DPAS recurrences can be easily adapted to tackle problems satisfying Composed DPAS recurrences.

### 5.3.3 Hybrid algorithm for Composed DPAS

The hybrid algorithm for Composed DPAS derives naturally from Algorithm 2. It amounts to replacing the recurrence (Add-DPAS), respectivelly (Add-D-DPAS), by (Comp-DPAS), respectivelly (Comp-D-DPAS), resulting in Algorithm 3. Eventually, we use Algorithm 3 as a subroutine to solve Equation (5.6), i.e. to solve the nominal problem $\mathcal{P}$.

**Algorithm 3:** Q-DDPAS for Composed DPAS

**Input:** $\epsilon_0 \in E$, auxiliary problem $P'$ satisfying (Comp-DPAS) and (Comp-D-DPAS)
**Output:** $\mathrm{OPT}[[n], 0, \epsilon_0]$ with high probability
**begin classical part**
  **for** $X \subseteq [n]$ *such that* $|X| = n/4$, *and* $t \in T$ **do**
    Compute $\mathrm{OPT}[X, t, \epsilon_0]$ with (Comp-DPAS) and store the results in the QRAM;
**begin quantum part**
  Apply Quantum Minimum Finding with (Comp-D-DPAS) to find $\mathrm{OPT}[[n], 0, \epsilon_0]$;
  To get values for the Quantum Minimum Finding above (the values $\mathrm{OPT}[J, t, \epsilon]$ for $J \subseteq [n]$ of size $n/2$, $t \in T$ and $\epsilon \in E$), apply Quantum Minimum Finding with (Comp-D-DPAS);
  To get values for the Quantum Minimum Finding above (the values $\mathrm{OPT}[X, t', \epsilon']$ for $X \subseteq [n]$ of size $n/4$, $t' \in T$ and $\epsilon' \in E$), get them on the QRAM

**Lemma 5.3.5.** *Let $\epsilon_0 \in E$. The bounded-error algorithm Q-DDPAS (Algorithm 3) solves $P'([n], 0, \epsilon_0)$ in $\mathcal{O}^*(|E|^2 \cdot |T| \cdot 1.754^n)$.*

Notice that the implementation of this algorithm is slightly different from the one of Algorithm 2, mainly due to the operation of composition. The details are given in Chapter 6.

**Algorithm 4:** Meta-algorithm with subroutine Q-DDPAS for Composed DPAS

**Input:** Auxiliary problem $P'$ satisfying (Comp-DPAS) and (Comp-D-DPAS)

**Output:** $\min_{\epsilon \in E}\left\{\epsilon : \mathrm{OPT}[[n], 0, \epsilon] < +\infty\right\}$ with high probability

```
ε* ← +∞;
for ε ∈ E do
    Solve P([n], 0, ε) with Algorithm 3;
    if OPT[[n], 0, ε] < +∞ and ε < ε* then
        ε* ← ε;
Return ε*
```

**Theorem 5.3.6.** *The bounded-error Algorithm 4, with Q-DDPAS as a subroutine, solves $\mathcal{P}$ in $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 1.754^n)$.*

As for the case of Q-DDPAS for Additive DPAS, we can reduce the exponential part of Q-DDPAS complexity for Composed DPAS, by the modification indicated in Observation 5.2.10, thus leading to the following observation.

**Observation 5.3.7.** *A slight modification of the Q-DDPAS algorithm can reduce the complexity of Algorithm 4 to $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 1.728^n)$.*

We illustrate in Section 5.4.2 the application of Q-DDPAS for Composed DPAS to the problem $1|r_j|\sum w_j U_j$ described in Section 5.3.1, together with another similar scheduling problem.

## 5.4 Application to the scheduling literature

In Section 5.2.2 and Section 5.3.2, we provided general formulations of problems satisfying Additive and Composed DPAS recurrences. Next, we illustrate these recurrences with several NP-hard single-machine scheduling problems enabling their resolution with our hybrid algorithm Q-DDPAS. The list of problems is non-exhaustive but highlights the structures' specificity of scheduling problems that enable such recurrences. Eventually, for each problem, we compare the worst-case time complexity of Q-DDPAS with the complexity of the best-known classical exact algorithm, which is a moderate exponential-time algorithm. Q-DDPAS improves the exponential-part complexity, sometimes at the cost of an additional pseudo-polynomial factor.

### 5.4.1 Scheduling with deadlines and precedence constraints

Single-machine scheduling problems with no constraints, deadline constraints or precedence constraints satisfy the *addition* of optimal values of the problem on sub-instances. We provide next several examples of problems that satisfy Additive DPAS and thus can be solved by Q-DDPAS (Algorithm 2).

In Subsection 5.2.1, we have presented the problem of minimizing the total weighted completion time with deadline constraints ($1|\tilde{d}_j|\sum_j w_j C_j$). The formulation needed the set $T$ to be equal

to $[\![0, \sum_{j=1}^{n} p_j]\!]$, hence its resolution with Q-DDPAS in $\mathcal{O}^*(\sum p_j \cdot 1.728^n)$ according to Observation 5.2.10. Next, we give two more examples, beginning with the strongly NP-hard scheduling problem with minimization of the total weighted tardiness. Henceforth, to lighten the equations, we note $p(J) = \sum_{j \in J} p_j$ the sum of processing times of the jobs in $J \subseteq [n]$.

**Example 12** (Minimizing the total weighted tardiness, $1||\sum_j w_j T_j$)**.** *For each job $j \in [n]$, we are given a weight $w_j$, a processing time $p_j$, and a due date $d_j$ that indicates the time after which the job is late. Thus, a job $j$ is late in permutation $\pi$ if its completion time is larger than $d_j$, and we define as $T_j(\pi) = \max(0, C_j(\pi) - d_j)$ its tardiness. Our problem aims at finding the permutation that minimizes the total weighted tardiness, referred to as $1||\sum_j w_j T_j$ in the scheduling literature. Let $T = [\![0, \sum_{j=1}^{n} p_j]\!]$ be the set of all possible starting times.*

*We define the related problem $P$ of Equation* (5.4) *as follows: for $J \subseteq [n]$ and $t \in T$,*

$$\Pi(J, t) = S_J \,,$$

*and for $\pi \in \Pi(J, t)$:*

$$f(\pi, J, t) = \sum_{j \in J} w_j \max(0, C_j(\pi) - d_j + t) \,,$$

*where* $\max(0, C_j - d_j + t)$ *represents the tardiness of job $j$ for the* effective due date $(d_j - t)$*. Problem $1||\sum_j w_j T_j$ satisfies both Additive DPAS recurrences. Indeed, Equation* (Add-DPAS) *is valid with: $\forall J \subseteq [n], \forall j \in J, \forall t \in T$,*

$$g(J, j, t) = w_j \max(0, p(J) - d_j + t) \,,$$

*where the computation of $g$ is polynomial (linear). Moreover, Equation* (Add-D-DPAS) *is valid for the following functions: $\forall X \subseteq J \subseteq [n]$ s.t. $|X| = |J|/2, \forall t \in T$,*

$$\tau_{shift}(J, X, t) = t + p(X) \qquad \text{and} \qquad h(J, X, t) = 0 \tag{5.7}$$

*initialized by, for $j \in [n]$ and $t \in T$,* $\mathrm{OPT}[\{j\}, t] = w_j \max(0, p_j - d_j + t)$ *.*

We consider the scheduling problem with precedence constraints and minimization of the total weighted completion time that is also NP-hard. Conversely to the two previous examples, the set $T$ is reduced to $\{0\}$, and function $h$ translates the potential infeasibility of the concatenation of problem $P$ on two sub-instances.

**Example 13** (Minimizing the total weighted completion time with precedence constraints, $1|prec|\sum_j w_j C_j$)**.** *We are given, for each job $j \in [n]$, a processing time $p_j$, a weight $w_j$, and a set of precedence constraints $K = \{(i, j) : i \prec j\}$. A pair of jobs $(i, j)$ in $K$ implies that $i$ must precede $j$ in the permutation, namely that $i$ must be processed before $j$. Our problem, denoted by $1|prec|\sum_j w_j C_j$, aims at finding the feasible permutation, i.e. that respects the precedence constraints, that minimizes the total weighted completion time. Let be $T = \{0\}$. Here, an instance of the problem $P$ of Equation* (5.4) *under consideration is only indexed by the chosen subset of $[n]$.*

*Thus, we consider the problem $P$ as follows: for $J \subseteq [n]$,*

$$\Pi(J, 0) = \{\pi \in S_J \,|\, \pi \text{ respects } K\} \,,$$

*and for $\pi \in \Pi(J, 0)$,*

$$f(\pi, J, 0) = \sum_{j \in J} w_j C_j(\pi) \,.$$

*Our problem* $1|prec|\sum_j w_jC_j$ *satisfies both Additive DPAS recurrences. Indeed, Equation* (Add-DPAS) *is valid for:*

$$\forall J \subseteq [n], \forall j \in J, \quad g(J,j,0) = \begin{cases} +\infty & \text{if } \exists (j,k) \in E | k \in J \\ w_j p(J) & \text{otherwise} \end{cases},$$

*where the computation of* $g$ *is polynomial (quadratic). This problem* $P$ *also satisfies* (Add-D-DPAS)*. Indeed, Equation* (Add-D-DPAS) *is valid for the following functions:* $\forall X \subseteq J \subseteq [n]$ *such that* $|X| = |J|/2$,

$$\tau_{shift}(J,X,0) = 0 \qquad \text{and} \qquad h(J,X,0) = \begin{cases} +\infty & \text{if } \exists (j,k) \in E | j \in J \setminus X \text{ and } k \in X \\ p(X) \cdot \sum\limits_{j \in J \setminus X} w_j & \text{otherwise} \end{cases}$$

*where the computation of* $h$ *is polynomial (quadratic). The initialization is, for* $j \in [n]$, $\mathrm{OPT}[\{j\},0] = w_j p_j$ .

The three NP-hard scheduling problems examples described above can be solved with Q-DDPAS for Additive DPAS (Algorithm 2). We illustrate in Table 5.1 the worst-case time complexities of solving them with Q-DDPAS and compare them with the complexities of the best-known exact classical algorithms. Q-DDPAS improves the complexity of the exponent but sometimes at the cost of a pseudo-polynomial factor.

| Problem | Q-DDPAS for Additive DPAS | Best classical algorithm |
|---|---|---|
| $1\|\tilde{d}_j\|\sum w_jC_j$ | $\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$ | $\mathcal{O}^*(2^n)$ (T'kindt et al., 2022) |
| $1\|\|\sum w_jT_j$ | $\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$ | $\mathcal{O}^*(2^n)$ (T'kindt et al., 2022) |
| $1\|prec\|\sum w_jC_j$ | $\mathcal{O}^*\left(1.728^n\right)$ | $\mathcal{O}^*((2-\epsilon)^n)$, for small $\epsilon$ (Cygan et al., 2014) |

Table 5.1: Comparison of complexities between Q-DDPAS and the best-known classical algorithms for several scheduling problems satisfying (Add-DPAS) and (Add-D-DPAS)

### 5.4.2 Scheduling with release date constraints

Single-machine scheduling problems with release date constraints do not satisfy the *addition* of optimal values of the problem on sub-instances but enable the *composition* of them. We illustrate this notion with two examples of problems that satisfy Composed DPAS and thus can be solved by Q-DDPAS (Algorithm 3).

We have presented in Subsection 5.3.1 an example that is the problem of minimizing the weighted number of late jobs with release date constraints ($1|r_j|\sum w_jU_j$). We have shown that the two sets to define the auxiliary problem are $E = [\![0, \sum_{j=1}^n w_j]\!]$ and $T = [\![0, \sum_{j=1}^n p_j]\!] \cup \{+\infty\}$. Thus, Q-DDPAS solves this problem in $\mathcal{O}^*\left((\sum w_j)^3 \cdot \sum p_j \cdot 1.728^n\right)$ according to Observation 5.3.7. Next, we present another example which is the strongly NP-hard problem of minimizing the total weighted completion time with release date constraints.

**Example 14** (Minimizing the total weighted completion time with release date constraints, $1|r_j|\sum w_jC_j$)**.** *Each job* $j \in [n]$ *has a weight* $w_j$*, a processing time* $p_j$*, and a release date* $r_j$*. This*

*problem aims at finding the feasible permutation, namely where each job starts after its release date, for which the total weighted completion time is minimal.*

*Let* $T = [\![0, \sum_{j=1}^{n} p_j]\!] \cup \{+\infty\}$ *and* $E = [\![0, \sum_{j=1}^{n} w_j \cdot \sum_{j=1}^{n} p_j]\!]$. *For a given* $\epsilon \in E$, *we consider the problem* $P'$ *of Equation* (5.5) *as follows:* $\forall J \subseteq [n], t \in T$,

$$P'(J,t,\epsilon): \quad \min_{\pi \in \Pi'(J,t,\epsilon)} C_{max}(\pi)\,,$$

*where* $C_{max}$ *is the is the maximum completion time, and*

$$\Pi'(J,t,\epsilon) = \{\pi \in S_J : C_j(\pi) \geq \max(t, r_j) + p_j \text{ and } \sum_{j \in J} w_j C_j(\pi) = \epsilon\}\,,$$

*where* $C_j$ *is the completion time of job* $j$.

*Problem* $P'$ *satisfies the two Composed DPAS recurrences* (Comp-DPAS) *and* (Comp-D-DPAS). *The initialization of the recurrences is, for* $j \in [n]$, $t \in T$ *and* $\epsilon \in E$,

$$\mathrm{OPT}[\{j\}, t, \epsilon] = \begin{cases} C_j := \max(t, r_j) + p_j, & \text{if } \epsilon = w_j C_j \\ +\infty, & \text{otherwise} \end{cases}$$

We synthesize in Table 5.2 the worst-case time complexities achieved by Q-DDPAS on the examples of scheduling problems satisfying the Composed DPAS recurrences. We compare them with the best-known classical complexities for exact algorithms. The latter comes from the algorithm of Inclusion-Exclusion designed by Ploton and T'kindt (2022), which provides a generic method to solve such problems. We observe that Q-DDPAS improves the exponential part of the complexity, at a cost of a higher degree for the pseudo-polynomial factor.

| Problem | Q-DDPAS for Composed DPAS | Best classical algorithm |
|---|---|---|
| $1\|r_j\|\sum w_j U_j$ | $\mathcal{O}^*\left((\sum w_j)^3 \cdot \sum p_j \cdot 1.728^n\right)$ | $\mathcal{O}^*(\sum w_j \cdot \sum p_j \cdot 2^n)$, (Ploton and T'kindt, 2022) |
| $1\|r_j\|\sum w_j C_j$ | $\mathcal{O}^*\left((\sum w_j)^3 \cdot (\sum p_j)^4 \cdot 1.728^n\right)$ | $\mathcal{O}^*(\sum w_j \cdot (\sum p_j)^2 \cdot 2^n)$, (Ploton and T'kindt, 2022) |

Table 5.2: Comparison of complexities between Q-DDPAS and the best-known classical algorithms for several scheduling problems satisfying (Comp-DPAS) and (Comp-D-DPAS)

## 5.5 Decision-based DPAS

We saw in the previous sections that the recurrence to solve $\mathcal{P}$ can be applied to a minimization problem, possibly involving an auxiliary problem. Sometimes, the recurrence does not apply directly to a minimization problem but to a *decision* problem. This is the case of the 3-machine flowshop problem. In this section, we adapt the hybrid algorithm Q-DDPAS to solve this problem. Notice that it easily generalizes to the $m$-flowshop problem, for $m \geq 4$.

### 5.5.1 3-machine flowshop and dynamic programming

We consider the permutation flowshop problem on 3 machines for $n$ jobs with minimizing the makespan as the objective function. This strongly NP-hard problem is often referred to as $F3||C_{\max}$ in the literature, and is still being actively studied, e.g. by Shang et al. (2018). Each job $j \in [n]$ consists of 3 operations $O_{ij}$ for $i \in [3]$, each operation being processed on the $i$-th

machine. We note $p_{ij}$ the processing time of operation $O_{ij}$. Each machine performs at most one operation at a time. For each job $j$, operations must be processed in the specific order $O_{1j}$, $O_{2j}$, $O_{3j}$: the first operation gets processed on the first machine, then the second operation gets processed on second machine (as soon as the first operation is finished and the second machine is available), and eventually the third operation gets processed on the third machine (as soon as the second operation is finished and the third machine is available). Thus, only the processing order of the jobs has to be decided, implying that a solution is entirely described by the permutation of jobs on the first machine. Thus, the problem can be formulated as

$$\min_{\pi \in S_{[n]}} C_{\max}(\pi)\,, \tag{5.8}$$

where $C_{\max}$ is the maximum completion time, called *makespan*, which is the completion time of the last job processed on the last machine (third machine). Because the two techniques presented so far do not apply to (5.8), we present an alternative approach involving the decision counterpart of the above optimization problem.

We introduce below a decision problem for deriving the recurrences. For that, we define the bounded set

$$T = \left[\!\!\left[ 0, \sum_{j\in[n], i\in[3]} p_{ij} \right]\!\!\right] \subseteq \mathbb{Z}\,.$$

**Definition 5.5.1** (Decision problem)**.** *For $J \subseteq [n]$, $\vec{\beta} = (\beta_2, \beta_3) \in T^2$ and $\vec{\epsilon} = (\epsilon_2, \epsilon_3) \in T^2$, we define the decision problem $D(J, \vec{\beta}, \vec{\epsilon})$ on a sub-instance associated with jobs in $J$ as the following question: "Does there exist a permutation $\pi \in S_J$ such that, for $i \in \{2, 3\}$, $b_i(\pi) \geq \beta_i$, and $e_i(\pi) \leq \epsilon_i$ ?", where $b_i(\pi)$, respectively $e_i(\pi)$, denotes the time at which the first operation begins, respectively the last operation ends, on the $i$-th machine.*

In other words, problem $D(J, \vec{\beta}, \vec{\epsilon})$ asks whether or not there exists a feasible permutation with jobs in $J$ such that it *holds* between the two *temporal fronts* $\vec{\beta}$ and $\vec{\epsilon}$. Notice that it is not necessary to impose any beginning and ending time for the first machine ($i = 1$). Indeed, the problem is time-invariant, thus we can always consider that the scheduling problem starts at time 0, and that the total completion time on the first machine is known and equal to the sum of processing times of the scheduled jobs. Notice that the number of parameters is four for the 3-machine flowshop, but generalizes to $2(m-1)$ parameters for the $m$-machine flowshop.

With these notations, $\mathcal{P}$ can be cast as follows:

$$\mathcal{P}: \qquad \min_{c\in T} \Big\{ c : D[[n], (0,0), (c,c)] = \text{True} \Big\}\,. \tag{5.9}$$

The decision problem $D$ satisfies both recurrences (Dec-DPAS) and (Dec-D-DPAS) below.

**Property 5.5.2** (Decision DPAS)**.** *For all $J \subseteq [n]$ of even cardinality, $\vec{\beta} \in T^2$ and $\vec{\epsilon} \in T^2$,*

$$D[J, \vec{\beta}, \vec{\epsilon}] = \bigvee_{\substack{X\subseteq J : |X| = |J|/2, \\ \vec{t}\in[\vec{\beta},\vec{\epsilon}]}} \Big( D[\{j\}, \vec{\beta}, \vec{t}] \wedge D[J \setminus \{j\}, \vec{t} \ominus p_{1j}, \vec{\epsilon} \ominus p_{1j}] \Big)\,, \tag{Dec-DPAS}$$

*where $\vec{t} \in [\vec{\beta}, \vec{\epsilon}]$ means that the $i$-th coordinate of $\vec{t}$ is between the $i$-th coordinates of $\vec{\beta}$ and $\vec{\epsilon}$, and where operation $\vec{v} \ominus c$, for a vector $\vec{v}$ and a constant $c$, subtracts $c$ to each coordinate of $\vec{v}$.*

This latter recurrence enables $\mathcal{P}$ to be solved by a classical dynamic programming algorithm.

**Lemma 5.5.3.** (Dec-DPAS) *solves* $\mathcal{P}$ *in* $\mathcal{O}^*(|T|^4 \cdot 2^n)$.

*Proof.* First, we can show that, for a given $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$, (Dec-DPAS) solves $D([n], \vec{\beta}_0, \vec{\epsilon}_0)$ in $\mathcal{O}^*(|T|^4 \cdot 2^n)$. This is essentially the same lines of the proof as in Lemma 5.2.2. Second, to solve $\mathcal{P}$, we make a dichotomic search over $T$ to find the minimum $c \in T$ such that $D([n], (0,0), (c,c))$ is True according to Equation (5.9). Thus, (Dec-DPAS) is called $\log_2(|T|)$ times. Because $|T| = \sum p_{ij}$ is a pseudo-polynomial term of the instance, the total complexity is

$$\mathcal{O}^*(\log_2(|T|) \cdot |T|^4 \cdot 2^n) = \mathcal{O}^*(|T|^4 \cdot 2^n)\,.$$

□

**Property 5.5.4** (Decision Dichotomic DPAS)**.** *For all* $J \subseteq [n]$ *of even cardinality,* $\vec{\beta} \in T^2$ *and* $\vec{\epsilon} \in T^2$*,*

$$D[J, \vec{\beta}, \vec{\epsilon}] = \bigvee_{\substack{X \subseteq J : |X| = |J|/2, \\ \vec{t} \in [\vec{\beta}, \vec{\epsilon}]}} \left( D[X, \vec{\beta}, \vec{t}] \wedge D[J \setminus X, \vec{t} \ominus \sum_{j \in X} p_{1j}, \vec{\epsilon} \ominus \sum_{j \in X} p_{1j}] \right). \qquad \text{(Dec-D-DPAS)}$$

**Lemma 5.5.5.** (Dec-D-DPAS) *solves* $\mathcal{P}$ *in* $\omega(|T|^4 \cdot 2^n)$.

*Proof.* This proof is similar to the proof of Lemma 5.2.5, with the argument that a dichotomic search is polynomial in the size of the instance as in the proof of Lemma 5.5.3. □

Once again, we observe that recurrence (Dec-DPAS) outperforms recurrence (Dec-D-DPAS) to solve by classical dynamic programming our problem $\mathcal{P}$. In the next section, we describe how we adapt Q-DDPAS to take advantage of those two recurrences to solve the 3-machine flowshop problem.

### 5.5.2 Hybrid algorithm for Decision-based DPAS

We call Q-Dec-DDPAS the adapted decision version of Q-DDPAS. The main difference is that instead of searching for a minimum value in a set in recurrence (Add-D-DPAS) or (Comp-D-DPAS), we search for a True value in a set in recurrence (Dec-D-DPAS). Thus, it essentially amounts to replacing Quantum Minimum Finding with the algorithm of Boyer et al. (1998) specified below, which extends Grover Search (Grover, 1996).

**Definition 5.5.6** (Grover Search Extension (Boyer et al., 1998))**.** *Let* $f : [n] \to \{0,1\}$ *be a function. Grover Search Extension computes with high probability the logical OR of all the* $f$ *values and the corresponding antecedent(s)* $x \in [n]$ *such that* $f(x) = 1$*. The complexity of Grover Search Extension is* $\mathcal{O}\left(\sqrt{n} \cdot C_f(n)\right)$*, where* $\mathcal{O}(C_f(n))$ *is the complexity of computing a value of* $f$*.*

Notice that we use Grover Search Extension instead of Grover Search because we do not know the number of $x$ such that $f(x) = 1$. The generalization by Boyer et al. (1998) enables us to deal with an unknown number of solutions while keeping the same complexity of Grover Search. Moreover, if there are $t \in \mathbb{N}^*$ solutions, the complexity is $\mathcal{O}\left(\sqrt{n/t} \cdot C_f(n)\right)$ but, having no bounds on $t$ whenever we call Grover Search Extension, we omit it in the complexity.

**Lemma 5.5.7.** *Let* $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$*. The bounded-error algorithm Q-Dec-DDPAS (Algorithm 5) solves* $D([n], \vec{\beta}_0, \vec{\epsilon}_0)$ *in* $\mathcal{O}^*((\sum p_{ij})^4 \cdot 1.754^n)$.

**Algorithm 5:** Q-Dec-DDPAS for 3-machine flowshop

**Input:** $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$, decision problem $D$ satisfying (Dec-DPAS) and (Dec-D-DPAS)
**Output:** $D[[n], \vec{\beta}_0, \vec{\epsilon}_0]$ with high probability
**begin classical part**
  **for** $X \subseteq [n] : |X| = n/4$ *and* $\vec{\beta}, \vec{\epsilon} \in T^2$ **do**
    Compute $D[X, \vec{\beta}, \vec{\epsilon}]$ with (Dec-DPAS) and store the results in the QRAM;
**begin quantum part**
  Apply Grover Search Extension with (Dec-D-DPAS) to find $D[[n], \vec{\beta}_0, \vec{\epsilon}_0]$;
  To get values for the Grover Search Extension above (the values $D[J, \vec{\beta}, \vec{\epsilon}]$ for $J \subseteq [n]$ of size $n/2$ and $\vec{\beta}, \vec{\epsilon} \in T$), apply Grover Search Extension with (Dec-D-DPAS);
  To get values for Grover Search Extension above (the values $D[X, \vec{\beta}', \vec{\epsilon}']$ for $X \subseteq [n]$ of size $n/4$ and $\vec{\beta}', \vec{\epsilon}' \in T$), get them on the QRAM;

*Proof.* As for the proof of Lemma 5.3.5, we follow the same reasoning of the proof of Theorem 5.2.8. The classical part computes and stores in the QRAM the decision variables for all $\binom{n/2}{n/4}$ bi-partitions of $[n]$, for any couple of parameters in $T^2$. The quantum part applies recursively twice Grover Search Extension. The first call puts the optimal values for sets of size $n/2$ in superposition. The second call finds the optimal value for all jobs in $[n]$.

The computation of the complexity in time of Q-Dec-DDPAS is also similar.

- Classical part: according to Lemma 5.5.3, (Dec-DPAS) computes all $D[X, \vec{\beta}, \vec{\epsilon}]$ for $X$ of size $n/4$ and $\vec{\beta}, \vec{\epsilon} \in T^2$ in

$$\mathcal{O}^*\left(|T|^4 \cdot \binom{n}{\leq n/4}\right) = \mathcal{O}^*(|T|^4 \cdot 2^{0.811n}) .$$

- Quantum part: the first call to Grover Search Extension *in parallel* is done on a set of size $|T|^2 \cdot \binom{n/2}{n/4}$. The second call to Grover Search Extension is done on a set of size $|T|^2 \cdot \binom{n}{n/2}$. Eventually, the complexity of the quantum part is:

$$\begin{aligned}\mathcal{O}^*\left(\sqrt{|T|^2 \cdot \binom{n/2}{n/4}}\sqrt{|T|^2 \cdot \binom{n}{n/2}}\right) &= \mathcal{O}^*\left(|T|^2\sqrt{\binom{n/2}{n/4}\binom{n}{n/2}}\right) \\ &= \mathcal{O}^*(|T|^2 \cdot 2^{0.75n}) .\end{aligned}$$

Eventually, the total complexity is

$$\mathcal{O}^*(|T|^4 \cdot 2^{0.811n} + |T|^2 \cdot 2^{0.75n}) = \mathcal{O}^*\left(|T|^4 \cdot 1.754^n\right) = \mathcal{O}^*\left((\sum p_{ij})^4 \cdot 1.754^n\right)$$

□

All the details of correctness and low-level implementation are given in Chapter 6.

**Theorem 5.5.8.** *The bounded-error Algorithm 6 solves the 3-machine flowshop in* $\mathcal{O}^*((\sum p_{ij})^4 \cdot 1.754^n)$ *with high probability.*

Once again, as mentioned in Observation 5.2.10, the complexity can be reduced thanks to a

**Algorithm 6:** Meta-algorithm with subroutine Q-Dec-DDPAS for the 3-machine flowshop

**Input:** 3-machine flowshop
**Output:** Minimum makespan with high probability

```
c* ← +∞;
for c ∈ T do
    Solve D([n], (0,0), (c,c)) with Algorithm 5;
    if D[[n], (0,0), (c,c)] = True and c < c* then
        c* ← c;
Return c*
```

slight modification on the Q-Dec-DDPAS that constitutes the subroutine, thus leading to the following observation.

**Observation 5.5.9.** *A slight modification of Algorithm 6 reduces the complexity of solving the 3-machine flowshop in* $\mathcal{O}^*((\sum p_{ij})^4 \cdot 1.728^n)$ *with high probability.*

This new method improves the best-known classical algorithm that is in $\mathcal{O}^*(3^n)$ or in $\mathcal{O}^*(M \cdot 2^n)$ if there exists a constant $M$ such that $p_{ij} \leq M$, for all $i \in [3], j \in [n]$, presented by Shang et al. (2018) and Ploton and T'kindt (2023). Hybrid bounded-error Algorithm 6 reduces the exponential part of the time complexity at the cost of a pseudo-polynomial factor. For most cases, this factor is negligible because the numerical values of 3-machine flowshop instances are small compared to the exponential part value. However, we present in the next subsection a way to dispose of this factor with an approximation scheme.

It is worth noting that the previous algorithm easily generalizes to the $m$-machine flowshop problem. Indeed, the only difference is the description of the *temporal front* that necessitates $2(m-1)$ parameters.

**Observation 5.5.10.** *The bounded-error Algorithm 6 generalizes to solve the* $m$*-machine flowshop in* $\mathcal{O}^*((\sum p_{ij})^{2(m-1)} \cdot 1.728^n)$ *with high probability.*

Notice that Ploton and T'kindt (2023) present a classical resolution for the $m$-machine flowshop by Inclusion-Exclusion in $\mathcal{O}^*((\sum p_{ij})^m \cdot 2^n)$.

### 5.5.3 Approximation scheme for the 3-machine flowshop

We present an approximation scheme for the 3-machine flowshop problem that trades the pseudo-polynomial factor in the complexity of Q-Dec-DDPAS and the optimality of the algorithm for a polynomial factor in $\frac{1}{\epsilon}$ and an approximation factor of $(1+\epsilon)$. In other words, we provide Algorithm 7 that finds a solution in time $\mathcal{O}^*\left(\frac{1}{\epsilon^3} \cdot 1.728^n\right)$ for which the makespan is not greater than $(1+\epsilon)$ times the optimal makespan. The latter point denotes that this is an $\epsilon$-approximation scheme. Our algorithm belongs to the class of moderate exponential-time approximation algorithms. Notice that the 3-machine flowshop problem does not admit an FPTAS (fully polynomial-time approximation scheme) because it is strongly NP-hard, meaning that no $\epsilon$-approximation algorithm exists to solve the 3-machine flowshop in time $\mathcal{O}\left(poly(n, \frac{1}{\epsilon})\right)$ unless P = NP (Vazirani, 2001). In comparison, Hall (1998) provides for the $m$-machine flowshop problem an FPT-AS (fixed-parameter tractable approximation scheme), namely an $\epsilon$-approximation algorithm that

runs in time $\mathcal{O}(f(\epsilon,\kappa)\cdot poly(n))$ for $\kappa$ a fixed parameter of the instance and $f$ a computable function. Hall (1998) choose $\kappa$ to be the number of machines of the flowshop, leading to an FPT-AS that runs in time $\mathcal{O}\left(n^{3.5}\cdot(\frac{m}{\epsilon})^{\frac{m^4}{\epsilon^2}}\right)$. In our case, we consider the case $m=3$.

**Algorithm 7:** Hybrid approximation scheme for the 3-machine flowshop

**Input:** $\epsilon > 0$, 3-machine flowshop on $n$ jobs with processing times $\{p_{ij} : i \in [3], j \in [n]\}$
**Output:** solution at most $1+\epsilon$ times the optimal solution

1 $P = \max_{i\in[3],j\in[n]}\{p_{ij}\}$;
2 $K = \frac{\epsilon P}{n+2}$;
3 **for** $i \in [3], j \in [n]$ **do**
4 $\quad p'_{ij} = \lceil \frac{p_{ij}}{K} \rceil$;
5 Solve the 3-machine flowshop on $n$ jobs with new processing times $\{p'_{ij} : i \in [3], j \in [n]\}$ with Algorithm 6 that outputs permutation $\pi'$;
6 Return $\pi'$

**Lemma 5.5.11.** *Let $\pi^*$ be an optimal solution of the 3-machine flowshop problem, for the processing times $\{p_{ij} : i \in [3], j \in [n]\}$. Let $\pi'$ be the output of Algorithm 7. We have*

$$C_{max}(\pi') \leq (1+\epsilon)\cdot C_{max}(\pi^*)\,.$$

Next, we introduce two observations necessary to prove Lemma 5.5.11. The proofs are omitted because of their simplicity.

**Observation 5.5.12.** *Let $\pi$ be a permutation and let $\alpha$ be a non-negative real number. We note $C_{max}(\pi)$ the makespan of $\pi$ of the 3-machine flowshop for processing times $\{p_{ij} : i \in [3], j \in [n]\}$. We note $C'_{max}(\pi)$ the makespan of $\pi$ of the 3-machine flowshop for processing times $\{p'_{ij} : i \in [3], j \in [n]\}$ such that $p'_{ij} := \alpha p_{ij}$ for all $i, j$. Then,*

$$C'_{max}(\pi) = \alpha C_{max}(\pi)\,.$$

*Notice that for $p'_{ij} \leq \alpha p_{ij}$, we have $C'_{max}(\pi) \leq \alpha C_{max}(\pi)$ even if the critical path in $\pi$ may differ to obtain $C_{max}$ and $C'_{max}$.*

**Observation 5.5.13.** *Let $\pi$ be a permutation and let $\beta$ be a real number such that $\beta \geq -\min_{i\in[3],j\in[n]}\{p_{ij}\}$. We note $C_{max}(\pi)$ the makespan of $\pi$ of the 3-machine flowshop for processing times $\{p_{ij} : i \in [3], j \in [n]\}$. We note $C''_{max}(\pi)$ the makespan of $\pi$ of the 3-machine flowshop for processing times $\{p''_{ij} : i \in [3], j \in [n]\}$ such that $p''_{ij} := p_{ij} + \beta$ for all $i \in [3], j \in [n]$. Then,*

$$C''_{max}(\pi) \leq C_{max}(\pi) + \beta(n+2)\,.$$

*Notice that for $p''_{ij} \leq p_{ij} + \beta$, we still have $C''_{max}(\pi) \leq C_{max}(\pi) + \beta(n+2)$ even if the critical path in $\pi$ may differ to obtain $C_{max}$ and $C''_{max}$.*

*Proof of Lemma 5.5.11.* Let be $\epsilon > 0$. The new processing times considered $p'_{ij} := \lceil \frac{p_{ij}}{K} \rceil$ imply that $\frac{p_{ij}}{K} \leq p'_{ij} < \frac{p_{ij}}{K} + 1$. We note $C'_{\max}$ the makespan of the new problem, i.e. the 3-machine flowshop problem with processing times $\{p'_{ij} : i \in [3], j \in [n]\}$.

On the one hand, we have $p'_{ij} < \frac{p_{ij}}{K} + 1$, for all $i \in [3], j \in [n]$. Thus, according to Observations 5.5.12 and 5.5.13 considering the optimal permutation $\pi^*$,

$$C'_{\max}(\pi^*) \leq \frac{C_{\max}(\pi^*)}{K} + n + 2\,,$$

namely, because $K > 0$,

$$KC'_{\max}(\pi^*) \leq C_{\max}(\pi^*) + K(n+2)\,. \tag{5.10}$$

On the other hand, we have $\frac{p_{ij}}{K} \leq p'_{ij}$. Thus, according to Observation 5.5.12 considering the output permutation $\pi'$ of Algorithm 7,

$$\frac{C_{\max}(\pi')}{K} \leq C'_{\max}(\pi')\,,$$

namely, because $K > 0$,

$$\begin{aligned} C_{\max}(\pi') &\leq KC'_{\max}(\pi') \leq KC'_{\max}(\pi^*) && (5.11)\\ &\leq C_{\max}(\pi^*) + K(n+2) = C_{\max}(\pi^*) + \epsilon P && (5.12)\\ &\leq C_{\max}(\pi^*) + \epsilon C_{\max}(\pi^*) = (1+\epsilon)C_{\max}(\pi^*)\,, && (5.13) \end{aligned}$$

where (5.11) comes from the fact that $\pi'$ is the optimal solution for makespan $C'_{\max}$, (5.12) results from Equation (5.10), and (5.13) is true because the makespan is always larger than $P = \max\limits_{i\in[3],j\in[n]}\{p_{ij}\}$. □

**Theorem 5.5.14.** *Algorithm 7 is an approximation scheme for the 3-machine flowshop problem and outputs a solution whose makespan is at most* $(1+\epsilon)$ *times the optimal value in time* $\mathcal{O}^*\left(\frac{1}{\epsilon^3} \cdot 1.728^n\right)$ .

*Proof.* First, according to Lemma 5.5.11, Algorithm 7 outputs a solution whose makespan is at most $(1+\epsilon)$ times the optimal value. Second, Algorithm 6 solves the new problem in time $\mathcal{O}^*((\sum p'_{ij})^4 \cdot 1.728^n) = \mathcal{O}^*(\frac{1}{\epsilon^4} \cdot 1.728^n)$. Indeed,

$$\begin{aligned} \sum p'_{ij} &\leq \sum \left(\frac{p_{ij}}{K} + 1\right)\\ &= \frac{1}{K}\sum p_{ij} + 3n\\ &\leq \frac{1}{K} \cdot 3nP + 3n\\ &= \frac{3n(n+2)}{\epsilon} + 3n\,. \end{aligned}$$

Thus, $\sum p'_{ij} \leq poly(n, \frac{1}{\epsilon})$. □

## 5.6 Conclusion

In this chapter, we proposed a hybrid algorithm Q-DDPAS in addition to generalized dynamic programming recurrences, to solve a broad class of NP-hard scheduling problems. Notice that our algorithm is an adapted version of the algorithm of Ambainis et al. (2019). Q-DDPAS, or its adaptation Q-Dec-DDPAS to decision problems, provides a quantum speed-up to their exact

resolution. Specifically, our hybrid algorithm reduces the best-known classical time complexity, often equal to $\mathcal{O}^*(2^n)$ for single-machine problems and $\mathcal{O}^*(3^n)$ for the 3-machine flowshop, to $\mathcal{O}^*(1.728^n)$, sometimes at the cost of an additional pseudo-polynomial factor as summarized in Table 5.3. We further discuss in Chapter 6 the details of the algorithms presented above.

| Problem | Our hybrid algorithm | Best classical algorithm |
|---|---|---|
| $1\|\tilde{d}_j\|\sum w_jC_j$ | $\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$ | $\mathcal{O}^*(2^n)$ (T'kindt et al., 2022) |
| $1\|\|\sum w_jT_j$ | $\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$ | $\mathcal{O}^*(2^n)$ (T'kindt et al., 2022) |
| $1\|prec\|\sum w_jC_j$ | $\mathcal{O}^*\left(1.728^n\right)$ | $\mathcal{O}^*((2-\epsilon)^n)$, for small $\epsilon$ (Cygan et al., 2014) |
| $1\|r_j\|\sum w_jU_j$ | $\mathcal{O}^*\left((\sum w_j)^3 \cdot \sum p_j \cdot 1.728^n\right)$ | $\mathcal{O}^*(\sum w_j \cdot \sum p_j \cdot 2^n)$ (Ploton and T'kindt, 2022) |
| $1\|r_j\|\sum w_jC_j$ | $\mathcal{O}^*\left((\sum w_j)^3 \cdot (\sum p_j)^4 \cdot 1.728^n\right)$ | $\mathcal{O}^*(\sum w_j \cdot (\sum p_j)^2 \cdot 2^n)$ (Ploton and T'kindt, 2022) |
| $F3\|\|C_{\max}$ | $\mathcal{O}^*\left((\sum p_{ij})^4 \cdot 1.728^n\right)$ | $\mathcal{O}^*(3^n)$ (Shang et al., 2018; Ploton and T'kindt, 2023) |

Table 5.3: Comparison of worst-case time complexities between our hybrid algorithm and the best-known classical algorithms.

# 6

# Low-level description of hybrid scheduling algorithms

This chapter is the continuation of the previous one: we provide a low-level description of the algorithms of Q-DDPAS and its decision-based version (Algorithm 2, Algorithm 3, and Algorithm 5). We call low-level description the description of the quantum part of the above-mentioned algorithms, with unitary circuits representing complete algorithms or single operations. This description aims to provide the necessary details to prove the correctness of these algorithms, and consequently enlighten possible implementations.

## 6.1 Preliminaries

Let us introduce the building blocks required for the description of the algorithms in this chapter.

### 6.1.1 Building block quantum circuits

We specify the quantum circuit associated with two algorithms that we use in a black box way and which constitute fundamental subroutines in Q-DDPAS or Q-Dec-DDPAS algorithms. The first one is the Quantum Minimum Finding algorithm of Dürr and Høyer (1996) presented in Definition 5.2.6, needed for Q-DDPAS. The second one is the Grover Search Extension of Boyer et al. (1998) presented in Definition 5.5.6, needed for Q-Dec-DDPAS. The circuit associated with the Quantum Minimum Finding is the following.

**Definition 6.1.1** (Circuit $U_{\text{QMF}}$)**.** *Let $f : [n] \to \mathbb{Z}$ be a function and let $U_f$ be its corresponding quantum circuit, specifically,*

$$U_f \left|i\right\rangle \left|0\right\rangle = \left|i\right\rangle \left|f(i)\right\rangle , \quad \forall i \in [n] .$$

*We note $U_{\text{QMF}}[U_f]$ the quantum circuit corresponding to the Quantum Minimum Finding algorithm of Dürr and Høyer (1996) that computes with high probability the minimum value of $f$ and the corresponding minimizer:*

$$U_{\text{QMF}}[U_f] \sum_{i=1}^{n} \frac{1}{\sqrt{n}} \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle = \sum_{i=1}^{n} \frac{1}{\sqrt{n}} \left|i\right\rangle \left|\arg\min_{i\in[n]}\{f(i)\}\right\rangle \left|\min_{i\in[n]}\{f(i)\}\right\rangle .$$

Next, we present the circuit associated with the extension of Grover Search.

**Definition 6.1.2** (Circuit $U_G$)**.** *Let $f : [n] \to \{0, 1\}$ be a function and let $U_f$ be its corresponding quantum circuit. We note $U_G[U_f]$ the quantum circuit corresponding to the algorithm of Boyer et al. (1998) that computes with high probability the logical OR of all the $f$ values. If it happens*

*to be True, $U_G[U_f]$ also gives the corresponding set $I_f = \{i : f(i) = 1\}$. Specifically,*

$$U_G[U_f] \sum_{i=1}^{N} \frac{1}{\sqrt{N}} |i\rangle |0\rangle |0\rangle = \sum_{i=1}^{N} \frac{1}{\sqrt{N}} |i\rangle |I_f\rangle \left| \bigvee_{i\in[N]} f(i) \right\rangle ,$$

Henceforth, we only look at the gate complexity of our algorithm. Thus, we deliberately ignore extra qubits required in Quantum Minimum Finding, Grover Search Extension, and classical computation as quantum circuits (see Observation 2.3.1).

### 6.1.2 Quantum circuits indexing

Before going into the description of the algorithms, we introduce some notations about indexing quantum circuits to be able to describe them rigorously. Let $reg = |q_1\rangle \dots |q_n\rangle$ be a register of $n$ qubits and $U$ be an operator acting on $k$ qubits, with $k < n$. Let $I$ be a $k$-tuple of distinct indices in $[n]$, $I = (i_1, \dots, i_k)$. We denote by $U^I$ the operator acting on the full register *reg*, that applies $U$ on $|q_{i_1}\rangle \dots |q_{i_k}\rangle$, and applies $Id$ on the remaining qubits. For instance, if $I$ is the tuple of contiguous indices $(3, \dots, k+3)$ with $k < n-3$, then

$$U^I = Id^{\otimes 2} \otimes U \otimes Id^{\otimes n-k-3} .$$

For $I = (i_1, \dots, i_k)$ and $J = (j_1, \dots, j_l)$ two distinct tuples in $[n]$ ($k$-tuple and $l$-tuple where $i \neq j, \forall (i,j) \in I \times J$) , we note $I \oplus J$ the concatenation of $I$ and $J$, namely $I \oplus J = (i_1, \dots, i_k, j_1, \dots, j_l)$. Regarding the Quantum Minimum Finding operator, let us denote the indexes related to the quantum circuit $U_f$ of a function $f$ as

$$U_f \underbrace{|i\rangle}_{I} \underbrace{|0\rangle}_{J} = \underbrace{|i\rangle}_{I} \underbrace{|f(i)\rangle}_{J} .$$

To clarify the computations detailed next, we index the corresponding Quantum Minimum Finding operator as $U_{\mathrm{QMF}}[U_f^I]$. We omit the index $J$ because this is an *auxiliary* register that does not appear in the output of $U_{\mathrm{QMF}}[U_f]$. Similarly, we index the corresponding Grover Search Extension operator as $U_G[U_f^I]$ omitting the index $J$.

## 6.2 Hybrid algorithm Q-DDPAS

In this section, we describe in the gate-based quantum computing model our algorithm Q-DDPAS that applies to any problem satisfying (Add-DPAS) and (Add-D-DPAS) or (Comp-DPAS) and (Comp-D-DPAS). We introduce in the two following subsections the sets and quantum circuits that constitute the building blocks of Q-DDPAS, and we provide for each of them their complexity. Depending on the tackled problem $\mathcal{P}$ solved by the hybrid algorithm, these sets, respectively quantum circuits, slightly differ whether the related problem $P$ satisfies (Add-DPAS) and (Add-D-DPAS), or the related auxiliary problem $P'$ satisfies (Comp-DPAS) and (Comp-D-DPAS).

### 6.2.1 Additive DPAS sets and quantum circuits

Let us begin with the sets and related quantum circuits useful to the description of Q-DDPAS for solving problems whose related problem $P$ satisfies recurrences (Add-DPAS) and (Add-D-DPAS).

We define two sets $\Lambda_{\text{add}}$ and $\Omega_{\text{add}}$ indexed by $(J,t)$ for $J \subseteq [n]$ and $t \in T$. Essentially, the set $\Lambda_{\text{add}}(J,t)$ contains all the possible balanced bi-partitions of $J$ and the associated parameter value of $\tau_{\text{shift}}$. The second set $\Omega_{\text{add}}(J,t)$ contains the optimal solutions for each bi-partition in $\Lambda_{\text{add}}(J,t)$.

**Definition 6.2.1** (Sets $\Lambda_{\text{add}}$ and $\Omega_{\text{add}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even and for $t \in T$, we define the set*

$$\Lambda_{add}(J,t) = \left\{ (X, t, J \setminus X, \tau_{shift}(J,X,t)) : X \subseteq J, |X| = \frac{|J|}{2} \right\},$$

*and the set*

$$\Omega_{add}(J,t) = \left\{ (X, \text{OPT}[X,t], J \setminus X, \text{OPT}[J \setminus X, \tau_{shift}(J,X,t)], t) : X \subseteq J, |X| = \frac{|J|}{2} \right\}.$$

The two following quantum circuits $U_{\Lambda_{\text{add}}}$ and $U_{\Omega_{\text{add}}}$ amount, respectively, to put into uniform superposition the elements of $\Lambda_{\text{add}}$ and $\Omega_{\text{add}}$.

**Definition 6.2.2** (Circuit $U_{\Lambda_{\text{add}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, and for $t \in T$, we define $U_{\Lambda_{add}}$ as follows:*

$$U_{\Lambda_{add}} |J\rangle |t\rangle |0\rangle^{\otimes 6} = |J\rangle |t\rangle \sum_{(\lambda_1^s, \lambda_1^t, \lambda_2^s, \lambda_2^t) \in \Lambda_{add}(J,t)} \frac{1}{\sqrt{|\Lambda_{add}(J,t)|}} |\lambda_1^s\rangle \left|\lambda_1^t\right\rangle |0\rangle |\lambda_2^s\rangle \left|\lambda_2^t\right\rangle |0\rangle .$$

Observe that we index the objects that represent sets by $s$, and the objects that represent scalars by $t$, because these are equal to the values in $T$.

**Proposition 6.2.3** (Complexity of $U_{\Lambda_{\text{add}}}$)**.** *The complexity of $U_{\Lambda_{add}}$ is polynomial in the size of the input.*

*Proof.* First, let us prove that, for a given $J \subseteq [n]$ of size $m$ for $m$ even, the construction of the quantum superposition of subsets of $J$ of size $m/2$ (i.e. superposition of balanced bi-partitions) is polynomial.

Let $J \subseteq [n]$ be of size $m$, for $m$ even. We note $\sigma_{\text{enum}} : J \mapsto [\![1, m]\!]$ the bijection that enumerates the elements of $J$. We note $\sigma_{\text{bipart}} : [\![1, \binom{m}{m/2}]\!] \mapsto \left\{ (A, [\![1,m]\!] \setminus A) : |A| = \frac{m}{2} \right\}$ the bijection that enumerates the balanced bi-partitions of $[\![1,m]\!]$. Let $U_{\sigma_{\text{bipart}}}$ be the quantum circuit corresponding to the function $\sigma_{\text{bipart}}$. Specifically, for $i \in [\![1, \binom{m}{m/2}]\!]$,

$$U_{\sigma_{\text{bipart}}} |i\rangle |0\rangle |0\rangle = |i\rangle \underbrace{|A_i\rangle |[\![1,m]\!] \setminus A_i\rangle}_{\sigma_{\text{bipart}}(i)} .$$

Let $U_{\sigma_{\text{enum}}^{-1}}$ be the quantum circuit corresponding to the inverse of the function $\sigma_{\text{enum}}$. Thus,

$$U_{\sigma_{\text{enum}}^{-1}} |i\rangle |A_i\rangle |[\![1,m]\!] \setminus A_i\rangle = |i\rangle |X_i\rangle |J \setminus X_i\rangle ,$$

for $X_i = \sigma_{\text{enum}}^{-1}(A_i) \subseteq J$. We denote by $\sigma_{\text{enum}}^{-1}(S)$, for $S$ a set, the operation of applying $\sigma_{\text{enum}}^{-1}$ to each element of $S$.

Consequently, we get a quantum superposition of all balanced bi-partitions of $J$ by applying first $U_{\sigma_{\text{bipart}}}$ then $U_{\sigma_{\text{enum}}^{-1}}$ to a quantum register that represents the superposition of all elements in $[\![1, \binom{m}{m/2}]\!]$. For that, we require $n_q := \lceil \log_2(\binom{m}{m/2}) \rceil = \mathcal{O}(m)$ qubits, each one initially in state

$|0\rangle$ on which we apply the Hadamard gate. Specifically,

$$\begin{aligned} U_{\sigma_{\text{enum}}^{-1}} U_{\sigma_{\text{bipart}}} H^{\otimes n_q} |0\rangle^{\otimes n_q} |0\rangle |0\rangle &= U_{\sigma_{\text{enum}}^{-1}} U_{\sigma_{\text{bipart}}} \sum_{i=1}^{\binom{m}{m/2}} |i\rangle |0\rangle |0\rangle \\ &= U_{\sigma_{\text{enum}}^{-1}} \sum_{i=1}^{\binom{m}{m/2}} |i\rangle |A_i\rangle |[\![1,m]\!] \setminus A_i\rangle \\ &= \sum_{i=1}^{\binom{m}{m/2}} |i\rangle |X_i\rangle |J \setminus X_i\rangle . \end{aligned}$$

Let us compute the complexity of $U_{\sigma_{\text{enum}}^{-1}} U_{\sigma_{\text{bipart}}} H^{\otimes n_q}$. For a given $i$, computing $\sigma_{\text{bipart}}(i)$, respectively $\sigma_{\text{enum}}^{-1}(i)$, is polynomial in $m$. According to Observation 2.3.1, the complexity of $U_{\sigma_{\text{bipart}}}$, respectively $U_{\sigma_{\text{enum}}^{-1}}$, is polynomial in $m$. Thus, the construction of the superposition of balanced bi-partitions of $J$ is polynomial.

Eventually, the computation of the function $\tau_{\text{shift}}$ is polynomial. Thus, the complexity of $U_{\Lambda_{\text{add}}}$ is polynomial. □

**Definition 6.2.4** (Circuit $U_{\Omega_{\text{add}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, and for $t \in T$, we define $U_{\Omega_{add}}$ as follows:*

$$U_{\Omega_{add}} |J\rangle |t\rangle |0\rangle = |J\rangle |t\rangle \sum_{\omega \in \Omega_{add}(J,t)} \frac{1}{\sqrt{|\Omega_{add}(J,t)|}} |\omega\rangle .$$

**Proposition 6.2.5** (Complexity of $U_{\Omega_{\text{add}}}$)**.** *Let $J$ be the input set. If we suppose to have stored in the QRAM the values $\text{OPT}[X,t]$ for all $X \subseteq J$ such that $|X| = |J|/2$ and for all $t \in T$, the complexity of $U_{\Omega_{add}}$ is polynomial in the size of the input.*

*Proof.* The proof follows essentially the same lines as the proof of Property 6.2.3. The quantum superposition of subsets is done in polynomial time, and instead of computing $\tau_{\text{shift}}$, we get values in the QRAM in constant time. □

We end this subsection with the definition of the quantum circuit of the addition required for recurrence (Add-D-DPAS).

**Definition 6.2.6** (Circuit $U_a$)**.** *We define the antecedent set $S_a = 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times T$. Let $a : S_a \to \mathbb{Z} \cup \{+\infty\}$ be the function:*

$$a(\omega_1^s, \omega_1^v, \omega_2^s, \omega_2^v, \omega^t) = \omega_1^v + \omega_2^v + h(\omega_1^s \cup \omega_2^s, \omega_1^s, \omega^t) .$$

*We note $U_a$ the quantum circuit corresponding to $a$, namely:*

$$\forall (\omega_1^s, \omega_1^v, \omega_2^s, \omega_2^v, \omega^t) \in S_a, \quad U_a |\omega\rangle |0\rangle = |\omega\rangle |a(\omega)\rangle ,$$

*where $|\omega\rangle = |\omega_1^s\rangle |\omega_1^v\rangle |\omega_2^s\rangle |\omega_2^v\rangle |\omega^t\rangle$ is encoded in five registers. Notice that we index the objects that represent numerical values by $v$.*

Notice that according to recurrence (Add-D-DPAS), the function $a$ applies on objects of $\Omega_{\text{add}}(J,t)$ for $J \subseteq [n]$ and $t \in T$, explaining the choice of the antecedent set.

**Proposition 6.2.7** (Complexity of $U_a$)**.** *The complexity of $U_a$ is polynomial in the size of the input.*

*Proof.* By assumption, the computation of $h$ is polynomial. It implies that the computation of $a$ is polynomial, and thus $U_a$ has a polynomial complexity (see Observation 2.3.1). □

**Remark 6.2.8.** *Notice that for $J \subseteq [n]$ and $t \in T$,*

$$\mathrm{OPT}[J,t] = \min_{\omega \in \Omega_{add}(J,t)} a(\omega)\,.$$

### 6.2.2 Composed DPAS sets and quantum circuits

In this subsection, we define the sets and their associated quantum circuits used for the description of the hybrid algorithm that solves problems whose related auxiliary problem satisfies recurrences (Comp-DPAS) and (Comp-D-DPAS). Similarly to the previous subsection, we define two sets $\Lambda_{\mathrm{comp}}$ and $\Omega_{\mathrm{comp}}$ indexed by $(J,t,\epsilon)$ for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$. In this case, the set $\Lambda_{\mathrm{comp}}(J,t,\epsilon)$ contains all the possible balanced bi-partitions of $J$ and the possible parameter values of $T$ and $E$. The second set $\Omega_{\mathrm{comp}}(J,t,\epsilon)$ contains the optimal solutions for each bi-partition and parameter values in $\Lambda_{\mathrm{comp}}(J,t,\epsilon)$.

**Definition 6.2.9** (Sets $\Lambda_{\mathrm{comp}}$ and $\Omega_{\mathrm{comp}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, for $t \in T$ and for $\epsilon \in E$, we define the set*

$$\Lambda_{comp}(J,t,\epsilon) = \left\{ (X, t_i, \epsilon_i, J \setminus X, t, \epsilon - \epsilon_i) : X \subseteq J, |X| = \frac{|J|}{2}, \epsilon_i \in E, t_i \in T \right\},$$

*and the set*

$$\Omega_{comp}(J,t,\epsilon) = \left\{ \begin{array}{r} (X, \mathrm{OPT}[X,t_i,\epsilon_i], t_i, \epsilon_i, J \setminus X, \mathrm{OPT}[J \setminus X, t, \epsilon - \epsilon_i], t, \epsilon - \epsilon_i) : \\ X \subseteq J, |X| = \frac{|J|}{2}, \epsilon_i \in E, t_i \in T \end{array} \right\}.$$

**Definition 6.2.10** (Circuit $U_{\Lambda_{\mathrm{comp}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, for $t \in T$ and for $\epsilon \in E$, we define $U_{\Lambda_{comp}}$ as follows:*

$$U_{\Lambda_{comp}} |J\rangle |t\rangle |0\rangle^{\otimes 8} = |J\rangle |t\rangle \sum_{\substack{(\lambda_1^s,\lambda_1^t,\lambda_1^e,\\ \lambda_2^s,\lambda_2^t,\lambda_2^e) \in \Lambda_{comp}(J,t,\epsilon)}} \frac{1}{\sqrt{|\Lambda_{comp}(J,t)|}} |\lambda_1^s\rangle \left|\lambda_1^t\right\rangle |\lambda_1^e\rangle |0\rangle |\lambda_2^s\rangle \left|\lambda_2^t\right\rangle |\lambda_2^e\rangle |0\rangle\,.$$

Observe that we index the objects that represent sets by $s$, the objects that represent scalars in $T$ by $t$, and the objects that represent parameter values in $E$ by $e$.

**Proposition 6.2.11** (Complexity of $U_{\Lambda_{\mathrm{comp}}}$)**.** *The complexity of $U_{\Lambda_{comp}}$ is polynomial in the size of the input.*

**Definition 6.2.12** (Circuit $U_{\Omega_{\mathrm{comp}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, for $t \in T$ and $\epsilon \in E$, we define $U_{\Omega_{comp}}$ as follows:*

$$U_{\Omega_{comp}} |J\rangle |t\rangle |\epsilon\rangle |0\rangle = |J\rangle |t\rangle |\epsilon\rangle \sum_{\omega \in \Omega_{comp}(J,t,\epsilon)} \frac{1}{\sqrt{|\Omega_{comp}(J,t,\epsilon)|}} |\omega\rangle\,.$$

**Proposition 6.2.13** (Complexity of $U_{\Omega_{\text{comp}}}$)**.** *Let $J$ be the input set. If we suppose to have stored in the QRAM the values $\text{OPT}[X,t,\epsilon]$ for all $X \subseteq J$ such that $|X| = |J|/2$, for all $t \in T$ and for all $\epsilon \in E$, the complexity of $U_{\Omega_{comp}}$ is polynomial in the size of the input.*

The proof of Proposition 6.2.11 (respectively Proposition 6.2.13) is similar to the proof of Proposition 6.2.3 (respectively Proposition 6.2.5).

The composition is the counterpart for (Comp-D-DPAS) of the addition for (Add-D-DPAS) (function $a$).

**Definition 6.2.14** (Circuit $U_c$)**.** *We note the antecedent set $S_c = 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times T \times E \times 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times T \times E$. Let $c : S_c \to \mathbb{Z} \cup \{+\infty\}$ be the function:*

$$c(\omega_1^s, \omega_1^v, \omega_1^t, \omega_1^e, \omega_2^s, \omega_2^v, \omega_2^t, \omega_2^e) = \begin{cases} \omega_1^v & \text{if } \omega_1^t = \omega_2^v \\ +\infty & \text{else} \end{cases}$$

*We note $U_c$ the quantum circuit corresponding to $c$, namely:*

$$\forall (\omega_1^s, \omega_1^v, \omega_1^t, \omega_1^e, \omega_2^s, \omega_2^v, \omega_2^t, \omega_2^e) \in S_c, \quad U_c \left|\omega\right\rangle \left|0\right\rangle = \left|\omega\right\rangle \left|c(\omega)\right\rangle \ ,$$

*where $\left|\omega\right\rangle = \left|\omega_1^s\right\rangle \left|\omega_1^v\right\rangle \left|\omega_1^t\right\rangle \left|\omega_1^e\right\rangle \left|\omega_2^s\right\rangle \left|\omega_2^v\right\rangle \left|\omega_2^t\right\rangle \left|\omega_2^e\right\rangle$ is encoded in eight registers.*

Notice that the function $c$ is meant to be applied on objects of $\Omega_{\text{comp}}(J,t,\epsilon)$, for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$, according to recurrence (Comp-D-DPAS).

**Proposition 6.2.15** (Complexity of $U_c$)**.** *The complexity of $U_c$ is polynomial in the size of the input.*

The proof of the above proposition is the same as for Proposition 6.2.7.

**Remark 6.2.16.** *Notice that, for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$,*

$$\text{OPT}[J,t,\epsilon] = \min_{\omega \in \Omega_{comp}(J,t,\epsilon)} c(\omega)\,.$$

### 6.2.3 Algorithm for Additive DPAS

Let us describe the hybrid algorithm Q-DDPAS in the gate-based quantum computing model. We begin with the description of Algorithm 2 which is Q-DDPAS for problems $\mathcal{P}$ whose related problem $P$ satisfies recurrences (Add-DPAS) and (Add-D-DPAS). Algorithm 3, which is Q-DDPAS for problems whose related auxiliary problem $P'$ satisfies recurrences (Comp-DPAS) and (Comp-D-DPAS), derives directly as we explain later in Subsection 6.2.4.

We present the quantum circuits used in the quantum part, as well as the numbering of the different registers.

- Let $|\text{ini}\rangle$ be the initial state:

$$|\text{ini}\rangle := \underbrace{|[n]\rangle\,|0\rangle}_{I^1} \underbrace{|0\rangle^{\otimes 3}}_{I^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \underbrace{|0\rangle^{\otimes 3}}_{I^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6},$$

  where the tuples indexing the different registers are decomposed as follows:

$$I^1 = I_1^1 \oplus I_2^1$$
$$I^2 = I_1^2 \oplus I_2^2 \oplus I_3^2$$

$$
\begin{aligned}
I^3 &= I_1^3 \oplus I_2^3 \\
I^4 &= I_1^4 \oplus I_2^4 \oplus I_3^4 \\
I^5 &= I_1^5 \oplus I_2^5 \\
I^6 &= I_1^6 \oplus I_2^6 .
\end{aligned}
$$

- Let
$$U_{\text{ini}} := (U_{\Omega_{\text{add}}}^{I^2} \otimes U_{\Omega_{\text{add}}}^{I^4}) \cdot U_{\Lambda_{\text{add}}}^{I^1 \oplus I^2 \oplus I^4} \tag{6.1}$$
be the quantum circuit that, given initial quantum state $|\text{ini}\rangle$, superposes all the couples $(X, X')$ such that $X, X' \subseteq [n]$, $|X| = |X'| = n/4$ and $X \cap X' = \emptyset$. For each couple, the optimal values and parameters associated are also superposed.

- The quantum circuit $U_{\text{QMF}}^{I_3^2 \oplus I^3}[U_a^{I_3^2}] \otimes U_{\text{QMF}}^{I_3^4 \oplus I^5}[U_a^{I_3^4}]$ applies two Quantum Minimum Finding *in parallel* (resulting from the tensor product of two quantum circuits) on the function $a$. Consequently, let
$$U_{\text{recur1}} := U_a^{I_1^2 \oplus I_2^3 \oplus I_1^4 \oplus I_2^5 \oplus I_2^1} \left( U_{\text{QMF}}^{I_3^2 \oplus I^3}[U_a^{I_3^2}] \otimes U_{\text{QMF}}^{I_3^4 \oplus I^5}[U_a^{I_3^4}] \right)$$
be the quantum circuit that adds, with the help of of function $a$, the resulting values of the two registers.

- Eventually, let
$$U_{\text{recur}} := U_{\text{QMF}}^{I_1^2 \oplus I_2^3 \oplus I_1^4 \oplus I_2^5 \oplus I_2^1 \oplus I^6}[U_{\text{recur1}}] \tag{6.2}$$
be the quantum circuit that applies Quantum Minimum Finding on the function represented by the circuit $U_{\text{recur1}}$.

We describe next the bounded-error hybrid algorithm Q-DDPAS (Algorithm 2) from a low-level point of view in Algorithm 8.

**Algorithm 8:** Q-DDPAS for Additive DPAS (low-level description)

**Input:** Problem $P$ satisfying (Add-DPAS) and (Add-D-DPAS)
**Output:** $\text{OPT}[[n], 0]$ with high probability
**begin classical part**
  **for** $X \subseteq [n] : |X| = n/4$ *and* $t \in T$ **do**
    Compute the optimal value $\text{OPT}[X, t]$ and the corresponding permutation $\pi^*[X, t]$ by classical (Add-DPAS);
    Store the tuple $(X, t, \text{OPT}[X, t], \pi^*[X, t])$ in the QRAM;
**begin quantum part**
  Prepare quantum state $|\text{ini}\rangle$;
  Apply the quantum circuit $U_{\text{recur}} U_{\text{ini}}$ to $|\text{ini}\rangle$;
  Measure register of indexes $I_2^6$;
Return the outcome of the measurement

We recall Theorem 5.2.8 which states Q-DDPAS worst-case time complexity. Because the complexity proof has already been proven in Chapter 5, we only provide the proof of the correctness, namely that the optimal value of $\mathcal{P}$ is stored in the register of indexes $I_2^6$ with high probability.

**Theorem 5.2.8.** *The bounded-error algorithm Q-DDPAS (Algorithm 2) solves $\mathcal{P}$ in $\mathcal{O}^*(|T| \cdot 1.754^n)$.*

*Proof.* Before entering the details of the computations, we give some intuition on the effect of the quantum circuit $U_{\text{recur}}U_{\text{ini}}$ and start by explaining the effect of $U_{\text{ini}}$ defined in Equation (6.1). First, the application of $U_{\Lambda_{\text{add}}}$ superposes all elements of $\Lambda_{\text{add}}([n], 0)$ in the registers of indexes $I^2$ (partition of $J$) and $I^4$ (partition of $[n] \setminus J$). This essentially amounts to superposing all the $\binom{n}{n/2}$ bi-partitions of $[n]$ where each sub-partition is of size $n/2$ (parameters $t$ included). Next, we apply $U_{\Omega_{\text{add}}}$ on register of index $I^2$, respectively $I^4$. This superposes all elements of $\Omega_{\text{add}}(J, t)$ (for a $J$ of size $n/2$ and $t \in T$ previously described in registers of indexes $I^2$, respectively $I^4$). This essentially amounts to superposing all the $\binom{n/2}{n/4}$ bi-partitions of $[n]$ where each sub-partition is of size $n/4$, parameters $t$ included, and the optimal value associated already stored in the QRAM.

Let us explain the effect of $U_{\text{recur}}$ defined in Equation (6.2). The application of $U_{\text{QMF}}[U_a]$ on a register encoding $(J,\ t)$ and the superposition of elements of $\Omega_{\text{add}}(J, t)$ stores $\text{OPT}[J, t]$ (with high probability) in an output register, according to Equation (Add-D-DPAS). Thus, $U_{\text{QMF}}[U_a]$ on register of index $I^2$, respectively $I^4$, superposes all $\text{OPT}[J, t]$ in $I^3$, respectively $I^5$, according to Remark 6.2.8. In other words, the circuit $U_{\text{QMF}}^{I_3^2 \oplus I^3}[U_a^{I_3^2 \oplus I_1^3}] \otimes U_{\text{QMF}}^{I_3^4 \oplus I^5}[U_a^{I_3^4 \oplus I_1^5}]$ that appears in $U_{\text{recur1}}$ superposes (with high probability) all optimal values of Equation (Add-D-DPAS) for $J$ of size $n/2$. Now that the optimal values are known for sets of size $n/2$ (before, we only knew optimal values for sets of size $n/4$), we apply one more time $U_{\text{QMF}}[U_a]$ on these new registers: it outputs $\text{OPT}[[n], 0]$ with high probability on the register of index $I_2^6$.

Next, we detail the computation of $U_{\text{recur}}U_{\text{ini}} |\text{ini}\rangle$ and show that $\text{OPT}[[n], 0]$ is stored in register of indexes $I_2^6$ with high probability. We write the following computations as if the algorithm Quantum Minimum Finding was returning the optimal solution with probability 1. First, we compute $U_{\text{ini}} |\text{ini}\rangle$.

$$\begin{aligned}
U_{\Lambda_{\text{add}}}^{I^1 \oplus I^2 \oplus I^4} |\text{ini}\rangle =& U_{\Lambda_{\text{add}}}^{I^1 \oplus I^2 \oplus I^4} \underbrace{|[n]\rangle}_{I^1} \underbrace{|0\rangle}_{I^2} \underbrace{|0\rangle^{\otimes 3}}_{I^3} \underbrace{|0\rangle^{\otimes 2}}_{I^4} \underbrace{|0\rangle^{\otimes 3}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6} \underbrace{|0\rangle^{\otimes 2}}_{I^6} \\
=& \underbrace{|[n]\rangle}_{I^1} \underbrace{|0\rangle}_{I^2} \sum_{(\lambda_1^s, \lambda_1^t, \lambda_2^s, \lambda_2^t) \in \Lambda_{\text{add}}([n],0)} \frac{1}{\sqrt{|\Lambda_{\text{add}}([n], 0)|}} \\
& \underbrace{|\lambda_1^s\rangle |\lambda_1^t\rangle |0\rangle}_{I^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \underbrace{|\lambda_2^s\rangle |\lambda_2^t\rangle |0\rangle}_{I^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6}.
\end{aligned}$$

Thus,

$$\begin{aligned}
U_{\text{ini}} |\text{ini}\rangle =& (U_{\Omega_{\text{add}}}^{I^2} \otimes U_{\Omega_{\text{add}}}^{I^4}) \cdot U_{\Lambda_{\text{add}}}^{I^1 \oplus I^2 \oplus I^4} |\text{ini}\rangle \\
=& (U_{\Omega_{\text{add}}}^{I^2} \otimes U_{\Omega_{\text{add}}}^{I^4}) \underbrace{|[n]\rangle}_{I^1} \underbrace{|0\rangle}_{I^2} \sum_{\substack{(\lambda_1^s, \lambda_1^t, \lambda_2^s, \lambda_2^t) \\ \in \Lambda_{\text{add}}([n],0)}} \frac{1}{\sqrt{|\Lambda_{\text{add}}([n], 0)|}} \underbrace{|\lambda_1^s\rangle |\lambda_1^t\rangle |0\rangle}_{I^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \underbrace{|\lambda_2^s\rangle |\lambda_2^t\rangle |0\rangle}_{I^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6} \\
=& \underbrace{|[n]\rangle}_{I^1} \underbrace{|0\rangle}_{I^2} \sum_{(\lambda_1^s, \lambda_1^t, \lambda_2^s, \lambda_2^t) \in \Lambda_{\text{add}}([n],0)} \frac{1}{\sqrt{|\Lambda_{\text{add}}([n], 0)|}} \underbrace{|\lambda_1^s\rangle}_{I_1^2} \underbrace{|\lambda_1^t\rangle}_{I_2^2} \left( \sum_{\omega \in \Omega_{\text{add}}(\lambda_1^s, \lambda_1^t)} \frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_1^s, \lambda_1^t)|}} \underbrace{|\omega\rangle}_{I_3^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \right) \\
& \underbrace{|\lambda_2^s\rangle}_{I_1^4} \underbrace{|\lambda_2^t\rangle}_{I_2^4} \left( \sum_{\omega \in \Omega_{\text{add}}(\lambda_2^s, \lambda_2^t)} \frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_2^s, \lambda_2^t)|}} \underbrace{|\omega\rangle}_{I_3^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \right) \underbrace{|0\rangle^{\otimes 2}}_{I^6}.
\end{aligned}$$

Second, we apply the tensor product of the two first Quantum Minimum Finding to the previous state.

$$
\begin{aligned}
&\left(U_{\mathrm{QMF}}^{I_3^2\oplus I^3}[U_a^{I_3^2}]\otimes U_{\mathrm{QMF}}^{I_3^4\oplus I^5}[U_a^{I_3^4}]\right)\underbrace{|[n]\rangle\,|0\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\mathrm{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\mathrm{add}}([n],0)|}}\\
&\underbrace{|\lambda_1^s\rangle}_{I_1^2}\underbrace{|\lambda_1^t\rangle}_{I_2^2}\left(\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{|\omega\rangle}_{I_3^2}\underbrace{|0\rangle^{\otimes 2}}_{I^3}\right)\\
&\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\left(\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\underbrace{|0\rangle^{\otimes 2}}_{I^5}\right)\underbrace{|0\rangle^{\otimes 2}}_{I^6}\\
=&\underbrace{|[n]\rangle\,|0\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\mathrm{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\mathrm{add}}([n],0)|}}\\
&\underbrace{|\lambda_1^s\rangle}_{I_1^2}\underbrace{|\lambda_1^t\rangle}_{I_2^2}\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{|\omega\rangle}_{I_3^2}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^3\otimes I_2^3}\\
&\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^5\otimes I_2^5}\underbrace{|0\rangle^{\otimes 2}}_{I^6}.
\end{aligned}
$$

Thus, we apply the second circuit of Quantum Minimum Finding.

$$
\begin{aligned}
U_{\mathrm{recur}}U_{\mathrm{ini}}\,|\mathrm{ini}\rangle =&U_{\mathrm{QMF}}^{I_1^2\oplus I_2^3\oplus I_1^4\oplus I_2^5\oplus I_2^1\oplus I^6}[U_{\mathrm{recur1}}]U_{\mathrm{ini}}\,|\mathrm{ini}\rangle\\
=&U_{\mathrm{QMF}}^{I_1^2\oplus I_2^3\oplus I_1^4\oplus I_2^5\oplus I_2^1 I^6}[U_a^{I_1^2\oplus I_2^3\oplus I_1^4\oplus I_2^5\oplus I_2^1}]\underbrace{|[n]\rangle\,|0\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\mathrm{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\mathrm{add}}([n],0)|}}\\
&\underbrace{|\lambda_1^s\rangle}_{I_1^2}\underbrace{|\lambda_1^t\rangle}_{I_2^2}\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{|\omega\rangle}_{I_3^2}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^3\otimes I_2^3}\\
&\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^5\otimes I_2^5}\underbrace{|0\rangle^{\otimes 2}}_{I^6}\\
=&\underbrace{|[n]\rangle\,|0\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\mathrm{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\mathrm{add}}([n],0)|}}\\
&\underbrace{|\lambda_1^s\rangle}_{I_1^2}\underbrace{|\lambda_1^t\rangle}_{I_2^2}\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{|\omega\rangle}_{I_3^2}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^3\otimes I_2^3}
\end{aligned}
$$

$$\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\sum_{\omega\in\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\underbrace{\left|\arg\min_{\omega} r(\omega)\right\rangle\left|\min_{\omega} r(\omega)\right\rangle}_{I_1^5\otimes I_2^5}$$

$$\underbrace{\left|\arg\min_{\lambda\in\Lambda_{\mathrm{add}}([n],0)} r(\lambda_1^s,\min_{\omega\in\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)} r(\omega),\lambda_2^s,\min_{\omega\in\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)} r(\omega),0)\right\rangle}_{I_1^6}$$

$$\underbrace{\left|\min_{\lambda\in\Lambda_{\mathrm{add}}([n],0)} r(\lambda_1^s,\min_{\omega\in\Omega_{\mathrm{add}}(\lambda_1^s,\lambda_1^t)} r(\omega),\lambda_2^s,\min_{\omega\in\Omega_{\mathrm{add}}(\lambda_2^s,\lambda_2^t)} r(\omega),0)\right\rangle}_{I_2^6}.$$

According to definition of $a$ and recurrence (Add-D-DPAS), the results stored in register of indexes $I_2^6$ is $\mathrm{OPT}[[n],0]$.

Notice that optimal permutation $\pi^*[[n],0]$ can be *rebuilt* with registers of indexes $I_1^3$, $I_1^5$ and $I_1^6$, and with the access to the results of the classical part in the QRAM. □

### 6.2.4 Adaptation for Composed DPAS

In this subsection, we adapt Algorithm 2, and consequently Algorithm 8, for a problem $\mathcal{P}$ related to auxiliary problem $P'$ satisfying recurrences (Comp-DPAS) and (Comp-D-DPAS). It essentially amounts to replacing $\Lambda_{\mathrm{add}}$ by $\Lambda_{\mathrm{comp}}$, $\Omega_{\mathrm{add}}$ by $\Omega_{\mathrm{comp}}$ and function $a$ by function $c$. Consequently, the quantum circuit $U_{\Lambda_{\mathrm{comp}}}$, respectively $U_{\Omega_{\mathrm{comp}}}$, apply on 8 registers, respectively 4 registers, that differ from Q-DDPAS for Additive DPAS. The resulting Algorithm 3 is provided in a low-level description in Algorithm 9.

Let us describe the slightly different quantum circuits adapting the number of registers and the registers on which they apply. Let $\epsilon_0 \in E$. The initial state is

$$|\mathrm{ini}\rangle = \underbrace{|[n]\rangle\,|0\rangle\,|\epsilon_0\rangle}_{I^1}\underbrace{|0\rangle^{\otimes 4}}_{I^2}\underbrace{|0\rangle^{\otimes 2}}_{I^3}\underbrace{|0\rangle^{\otimes 4}}_{I^4}\underbrace{|0\rangle^{\otimes 2}}_{I^5}\underbrace{|0\rangle^{\otimes 2}}_{I^6},$$

where the tuples indexing the different registers are decomposed as follows:

$$\begin{aligned}
I^1 &= I_1^1 \oplus I_2^1 \oplus I_3^1\\
I^2 &= I_1^2 \oplus I_2^2 \oplus I_3^2 \oplus I_4^2\\
I^3 &= I_1^3 \oplus I_2^3\\
I^4 &= I_1^4 \oplus I_2^4 \oplus I_3^4 \oplus I_4^4\\
I^5 &= I_1^5 \oplus I_2^5\\
I^6 &= I_1^6 \oplus I_2^6\,.
\end{aligned}$$

The three quantum circuits that appear on the quantum part are:

$$U_{\mathrm{ini}} = (U_{\Omega_{\mathrm{comp}}}^{I^2} \otimes U_{\Omega_{\mathrm{comp}}}^{I^4}) \cdot U_{\Lambda_{\mathrm{comp}}}^{I^1\oplus I^2\oplus I^4},$$

$$U_{\mathrm{recur1}} = U_c^{I_1^2\oplus I_2^3\oplus I_2^2\oplus I_3^2\oplus I_1^4\oplus I_2^5\oplus I_2^4\oplus I_3^4}\left(U_{\mathrm{QMF}}^{I_4^2\oplus I^3}[U_c^{I_4^2}]\otimes U_{\mathrm{QMF}}^{I_4^4\oplus I^5}[U_c^{I_4^4}]\right),$$

$$U_{\text{recur}} = U_{\text{QMF}}^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4 \oplus I^6}[U_{\text{recur1}}] \,.$$

Next, we describe with a low level of details Algorithm 3 which is the adaptation of Q-DDPAS to solve $P'([n], 0, \epsilon_0)$ for a given $\epsilon_0 \in E$.

**Algorithm 9:** Q-DDPAS for Composed DPAS (low-level description)

**Input:** $\epsilon_0 \in E$, auxiliary problem $P'$ satisfying (Comp-DPAS) and (Comp-D-DPAS)
**Output:** $\text{OPT}[[n], 0, \epsilon_0]$ with high probability
**begin classical part**
  **for** $X \subseteq [n] : |X| = n/4$ *and* $t \in T$ **do**
    Compute the optimal value $\text{OPT}[X, t, \epsilon_0]$ and the corresponding permutation $\pi^*[X, t, \epsilon_0]$ by classical (Comp-DPAS);
    Store the tuple $(X, t, \text{OPT}[X, t, \epsilon_0], \pi^*[X, t, \epsilon_0])$ in the QRAM;
**begin quantum part**
  Prepare quantum state $|\text{ini}\rangle$;
  Apply the quantum circuit $U_{\text{recur}} U_{\text{ini}}$ to $|\text{ini}\rangle$;
  Measure register of indexes $I_2^6$;
Return the outcome of the measurement

The proof of correctness of Lemma 5.3.5 is the same as for Theorem 5.2.8. To lighten the reading, and because the approach is very similar, we do not detail the calculations here.

## 6.3 Decision-based hybrid algorithm Q-Dec-DPAS

In what follows, we define the sets and their associated quantum circuits to describe the Q-Dec-DDPAS (Algorithm 5).

**Definition 6.3.1** (Sets $\Lambda_{\text{dec}}$ and $\Omega_{\text{dec}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even and for $\vec{\beta}, \vec{\epsilon} \in T^2$, we define the set*

$$\Lambda_{dec}(J, \vec{\beta}, \vec{\epsilon}) = \left\{ (X, \vec{\beta}, \vec{t}, J \setminus X, \vec{t}, \vec{\epsilon}) : X \subseteq J, |X| = \frac{|J|}{2}, \vec{t} \in [\vec{\beta}, \vec{\epsilon}] \right\} ,$$

*and the set*

$$\Omega_{dec}(J, \vec{\beta}, \vec{\epsilon}) = \left\{ (X, D[X, \vec{\beta}, \vec{t}], \vec{\beta}, \vec{t}, J \setminus X, D[J \setminus X, \vec{t}, \vec{\epsilon}], \vec{t}, \vec{\epsilon}) : X \subseteq J, |X| = \frac{|J|}{2}, \vec{t} \in [\vec{\beta}, \vec{\epsilon}] \right\} .$$

The quantum circuits associated with these two sets are the following.

**Definition 6.3.2** (Circuit $U_{\Lambda_{\text{dec}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, and for $\vec{\beta}, \vec{\epsilon} \in T^2$, we define $U_{\Lambda_{dec}}$ as follows:*

$$U_{\Lambda_{dec}} |J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle |0\rangle^{\otimes 8} =$$
$$|J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle \sum_{\substack{(\lambda_1^s, \lambda_1^{tb}, \lambda_1^{te}, \lambda_2^s, \lambda_2^{tb}, \lambda_2^{te}) \\ \in \Lambda_{dec}(J, \vec{\beta}, \vec{\epsilon})}} \frac{1}{\sqrt{|\Lambda_{dec}(J, \vec{\beta}, \vec{\epsilon})|}} |\lambda_1^s\rangle \left|\lambda_1^{tb}\right\rangle |\lambda_1^{te}\rangle |0\rangle |\lambda_2^s\rangle \left|\lambda_2^{tb}\right\rangle |\lambda_2^{te}\rangle |0\rangle \,.$$

*Notice that we index the objects that represent sets by s, and the objects that represent scalars in $T^2$ by tb if it represents a couple of beginning times, or by te if it represents a couple of ending times.*

**Proposition 6.3.3** (Complexity of $U_{\Lambda_{\text{dec}}}$)**.** *The complexity of $U_{\Lambda_{dec}}$ is polynomial in the size of the input.*

**Definition 6.3.4** (Circuit $U_{\Omega_{\text{dec}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, and for $\vec{\beta}, \vec{\epsilon} \in T^2$, we define $U_{\Omega_{dec}}$ as follows:*

$$U_{\Omega_{dec}} |J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle |0\rangle = |J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle \sum_{\omega \in \Omega_{dec}(J,\vec{\beta},\vec{\epsilon})} \frac{1}{\sqrt{|\Omega_{dec}(J,\vec{\beta},\vec{\epsilon})|}} |\omega\rangle \ .$$

**Proposition 6.3.5** (Complexity of $U_{\Omega_{\text{dec}}}$)**.** *Let $J$ be the input set. If we suppose to have stored in the QRAM the values $D[X, \vec{\beta}, \vec{\epsilon}]$ for all $X \subseteq J$ such that $|X| = |J|/2$ and for all $\vec{\beta}, \vec{\epsilon} \in T^2$, the complexity of $U_{\Omega_{dec}}$ is polynomial in the size of the input.*

The proof of Proposition 6.3.3, respectively Proposition 6.3.5, is similar to the proof of Proposition 6.2.3, respectively Proposition 6.2.5. Notice that $\vec{t} \in [\vec{\beta}, \vec{\epsilon}]$ can be replaced by $\vec{t} \in T^2$ in sets $U_{\Lambda_{\text{dec}}}$ and $U_{\Omega_{\text{dec}}}$ so that the circuits that superpose all elements of these sets are easier to design. Indeed, (Dec-DPAS) and (Dec-D-DPAS) are less accurate but still valid with this replacement.

The operation in recurrence (Dec-D-DPAS) is not the addition (represented by the function $a$ for (Add-D-DPAS)) nor the composition (represented by the function $c$ for (Comp-D-DPAS)) but the logical AND. We define below its corresponding quantum circuit.

**Definition 6.3.6** (Circuit $U_{and}$)**.** *We note the antecedent set $S_{and} = 2^{[n]} \times \{0,1\} \times T^2 \times T^2 \times 2^{[n]} \times \{0,1\} \times T^2 \times T^2$. Let $and : S_{and} \to \{0,1\}$ be the function:*

$$and(\omega_1^s, \omega_1^b, \omega_1^{tb}, \omega_1^{te}, \omega_2^s, \omega_2^b, \omega_2^{tb}, \omega_2^{te}) = \begin{cases} 1 & \text{if } \omega_1^b = \omega_2^b = 1 \\ 0 & \text{else} \end{cases}$$

*We note $U_{and}$ the quantum circuit associated to the function, specifically,*

$$\forall \omega = (\omega_1^s, \omega_1^b, \omega_1^{tb}, \omega_1^{te}, \omega_2^s, \omega_2^b, \omega_2^{tb}, \omega_2^{te}) \in S_{and}, \quad U_{and} |\omega\rangle |0\rangle = |\omega\rangle |and(\omega)\rangle \ .$$

Notice that objects representing boolean values are indexed by $b$. Note that according to recurrence (Dec-D-DPAS), the function *and* applies on objects of sets $\Omega_{\text{dec}}(J, \vec{\beta}, \vec{\epsilon})$ for $J \subseteq [n]$ and $\vec{\beta}, \vec{\epsilon} \in T^2$.

**Proposition 6.3.7** (Complexity of $U_{and}$)**.** *The complexity of $U_{and}$ is polynomial in the size of the input.*

The proof of the above proposition is the same as the one of Proposition 6.2.7.

**Remark 6.3.8.** *Notice that for $J \subseteq [n]$ and $\vec{\beta}, \vec{\epsilon} \in T^2$,*

$$D[J, \vec{\beta}, \vec{\epsilon}] = \bigvee_{\omega \in \Omega_{dec}(J,\vec{\beta},\vec{\epsilon})} and(\omega) \, .$$

Next, we describe the different quantum circuits for the quantum part of Q-Dec-DDPAS (Algorithm 5). Let $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$. The initial state is

$$|\text{ini}\rangle = \underbrace{|[n]\rangle \left|\vec{\beta}_0\right\rangle |\vec{\epsilon}_0\rangle}_{I^1} \underbrace{|0\rangle^{\otimes 4}}_{I^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \underbrace{|0\rangle^{\otimes 4}}_{I^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6},$$

where the tuples indexing the different registers are decomposed as follows:

$$\begin{aligned}
I^1 &= I_1^1 \oplus I_2^1 \oplus I_3^1 \\
I^2 &= I_1^2 \oplus I_2^2 \oplus I_3^2 \oplus I_4^2 \\
I^3 &= I_1^3 \oplus I_2^3 \\
I^4 &= I_1^4 \oplus I_2^4 \oplus I_3^4 \oplus I_4^4 \\
I^5 &= I_1^5 \oplus I_2^5 \\
I^6 &= I_1^6 \oplus I_2^6
\end{aligned}$$

The three quantum circuits that appear on the quantum part are:

$$U_{\text{ini}} = (U_{\Omega_{\text{dec}}}^{I^2} \otimes U_{\Omega_{\text{dec}}}^{I^4}) \cdot U_{\Lambda_{\text{dec}}}^{I^1 \oplus I^2 \oplus I^4},$$

$$U_{\text{recur1}} = U_{and}^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4} \left( U_G^{I_4^2 \oplus I^3}[U_{and}^{I_4^2}] \otimes U_G^{I_4^4 \oplus I^5}[U_{and}^{I_4^4}] \right),$$

$$U_{\text{recur}} = U_G^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4 \oplus I_6}[U_{\text{recur1}}].$$

We provide in Algorithm 10 the description of Algorithm 5 with the details about the quantum circuits.

**Algorithm 10:** Q-Dec-DDPAS for 3-machine flowshop (low-level description)

**Input:** $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$, decision problem $D$ satisfying (Dec-DPAS) and (Dec-D-DPAS)
**Output:** $D[[n], \vec{\beta}_0, \vec{\epsilon}_0]$ with high probability
**begin classical part**
  **for** $X \subseteq [n] : |X| = n/4$ *and* $\vec{\beta}, \vec{\epsilon} \in T^2$ **do**
    Compute the optimal value $D[X, \vec{\beta}, \vec{\epsilon}]$ and the corresponding permutation $\pi^*[X, \vec{\beta}, \vec{\epsilon}]$ by classical (Dec-DPAS);
    Store the tuple $(X, \vec{\beta}, \vec{\epsilon}, D[X, \vec{\beta}, \vec{\epsilon}], \pi^*[X, \vec{\beta}, \vec{\epsilon}])$ in the QRAM;
**begin quantum part**
  Prepare quantum state $|\text{ini}\rangle$;
  Apply the quantum circuit $U_{\text{recur}} U_{\text{ini}}$ to $|\text{ini}\rangle$;
  Measure register of indexes $I_2^6$;
Return the outcome of the measurement

Similarly to Q-DDPAS for the Additive or the Composed version, the correctness of Lemma 5.5.7 can be verified by the same type of computations.

## 6.4 Conclusion

In this chapter, we provided a low-level description of the hybrid algorithms tackling NP-hard scheduling problems of Chapter 5. This way, we provided proof of the correctness of these algorithms and proposed a way to implement them on gate-based quantum computers.

# 7

# Quantum robust optimization

Robust optimization is essential to address real-world optimization problems that involve inherent uncertainty. These practical applications can be found in various domains like transportation and logistics, healthcare, or telecommunications, where the problems deal with uncertain input data. Uncertainty can arise from factors such as traffic congestion, weather conditions, fluctuations in demand and resource availability, network equipment failures etc. For the railway domain in particular, uncertainty occurs at two levels. The first level relates to the category of *pre-operational* problems. For instance, timetabling planning must take into account the variations in travel times or station stop times, and ensure to be robust under these variations. The main reasons for the fluctuations are weather conditions or passengers' exchange increasing the dwell time. The second level concerns the category of *real-time* problems. These involve re-planning due to incidents, such as a catenary rupture or a train breakdown. In this case, the duration of the disruption can vary and is not known in advance. Mathematically, the uncertainty is modeled by unknown parameters that can take values from a set called uncertainty set, which is often discrete, polyhedral, or ellipsoidal, and contains all the possible/likely values for these parameters. The robustness of the solution comes from the fact that the model considers the worst-case scenario over the uncertainty set.

In this chapter, we study the use of quantum subroutines to tackle robust combinatorial optimization problems. Specifically, we consider the (MIN-MAX) problem, defined below, for which the uncertainty set is a polytope. Considering the classical algorithm of Omer et al. (2024) dedicated to solving this problem, we apply the two following quantum algorithms. The first one is Quantum Minimum Finding (Dürr and Høyer, 1996) and achieves a quadratic speedup. The second one is the Quantum Linear System Algorithm (Childs et al., 2017) and does not seem to provide any speedup under reasonable assumptions because of the numerical error generated by this subroutine. The first algorithm applies generally to various robust problems like the one of Bertsimas and Sim (2003) where we consider budgeted uncertainty sets. The second is specific to cases where the uncertainty set is a polytope because it involves solving linear systems. Such an application is notably used by Nannicini (2024) to provide fast quantum subroutines for the simplex method.

## 7.1 Robust optimization with polyhedral uncertainty

Let $n \in \mathbb{N}$ and let $\mathcal{Y} \subseteq \{0,1\}^n$. Let us consider the nominal problem

$$\min_{y \in \mathcal{Y}} \sum_{i \in [n]} c_i y_i \,, \tag{7.1}$$

where $c \in \mathbb{R}^n$ is the cost vector, and $\mathcal{Y}$ describes the set of feasible solutions. For instance, one can think of usual feasible sets such as the one containing spanning trees for the minimum spanning tree problem, or paths for the shortest path problem as considered in the robust optimization

of (Pugliese et al., 2019) for example. The robust counterpart of this problem,

$$\min_{y\in\mathcal{Y}} \max_{c\in U} \sum_{i\in[n]} c_i y_i \,,$$

for which only the cost vector is uncertain and described by a general polytope $U$, is strongly NP-hard even if the nominal problem is tractable (Buchheim and Kurtz, 2018). However, for specific polytopes, the tractability of the nominal problem can be preserved under some assumptions. This is the case for the polytope we consider in this chapter.

Let us consider the robust counterpart of problem (7.1), where the cost vector $c$ is uncertain and for which the positive deviation is in a polytope described by $s \in \mathbb{N}$ inequalities. Specifically, we consider the problem

$$\min_{y\in\mathcal{Y}} \max_{\xi\in\Xi} \sum_{i\in[n]} (c_i + \xi_i) y_i \,, \tag{MIN-MAX}$$

where the uncertainty set is the polytope

$$\Xi = \{\xi \in \mathbb{R}^n : A\xi \leq r,\, 0 \leq \xi \leq e\} \,,$$

for $A \in \mathbb{R}^{s\times n}$, $r \in \mathbb{R}^s$ and $e \in \mathbb{R}^n_+$. Notice that this general formulation covers the emblematic case of robust optimization with budgeted uncertainty first introduced by Bertsimas and Sim (2003), or its generalization to robust optimization with knapsack uncertainty (Poss, 2018). Next, we state a result from Omer et al. (2024) that can be summarized as follows: solving (MIN-MAX) amounts to solving $\mathcal{O}(n^s)$ nominal problems whose objective function is expressed according to the solutions of linear subsystems depending on $A$. Specifically, let $\Theta \subseteq \mathbb{R}^s_+$ be the set of all non-negative solutions of subsystems formed by $s$ linearly independent rows of

$$\begin{pmatrix} A^T \\ A^T \\ Id_{s\times s} \end{pmatrix} \theta = \begin{pmatrix} 1_n \\ 0_n \\ 0_s \end{pmatrix} . \tag{$S$}$$

For $L \subseteq \{1,\ldots,2n+s\}$ such that $|L| = s$, we note

$$G(L)\theta = h(L) \tag{$S_L$}$$

the subsystem of $(S)$ formed by the $s$ rows indexed by $L$. Henceforth, we note $[\alpha]^+$ the positive part of $\alpha \in \mathbb{R}$, namely $[\alpha]^+ = \max(0, \alpha)$.

**Theorem 7.1.1** (Omer et al. (2024))**.** *Solving* (MIN-MAX) *amounts to solving, for all $\theta \in \Theta$, the problem $(P_\theta)$ defined as follows:*

$$\min_{y\in\mathcal{Y}} g_\theta(y) = \sum_{k\in[s]} r_k\theta_k + (c + \beta_\theta)^T y \,, \tag{$P_\theta$}$$

*where the $i$-th coordinate of $\beta_\theta$ is*

$$\beta_{\theta,i} = \left[1 - \sum_{k\in[s]} a_{ki}\theta_k\right]^+ - \left[-\sum_{k\in[s]} a_{ki}\theta_k\right]^+ .$$

*In other words, if we note $u^*$ the optimal solution of* (MIN-MAX) *and $z^*_\theta$ the optimal solution of $(P_\theta)$,*

$$u^* = \min_{\theta\in\Theta} z^*_\theta \,. \tag{7.2}$$

Let $\mathcal{A}_\Theta$ be the classical algorithm that computes the expression of $g_\theta$, i.e. that solves a subsystem of $s$ rows and $s$ variables and outputs a vector (of $n$ components) of summaries statistic of the subsystem's solution. We designate by *summary statistic* the term $\sum_{k\in[s]} \alpha_k\theta_k$ for $\alpha \in \mathbb{R}^s$ and $\theta$ the subsystem's solution. We note $g(s,n)$ the time complexity of $\mathcal{A}_\Theta$. Notice that using the Gaussian elimination to solve the linear system results in the complexity $g(s,n) = \mathcal{O}(s^3+n)$. For sparse matrix, Yuster and Zwick (2005) uses fast matrix multiplication leading to the complexity $g(s,n) = \mathcal{O}(d^{0.7}s^{1.9} + s^{2+o(1)} + n)$, where $d$ is the sparsity of the matrix of the linear system.

Let $\mathcal{A}_P$ be a classical algorithm that solves problem $(P_\theta)$. We note $f(n)$ its time complexity.

**Corollary 7.1.2** (Omer et al. (2024))**.** *The number of subsystems of* $(S)$ *composed of* $s$ *linearly independent rows is no more than* $\mathcal{O}(n^s)$*. Thus, the complexity to solve* (MIN-MAX) *is* $\mathcal{O}(n^s(g(s,n)+f(n)))$*.*

It results that, for $s$ independent on $n$, the tractability of the nominal problem implies the tractability of its robust counterpart.

In Section 7.2, we describe a quantum-classical algorithm that amounts to apply Quantum Minimum Finding (see Definition 5.2.6) to solve the minimization problem (7.2) and thus reduce the above classical complexity. In Section 7.3, we experiment the use of an additional quantum subroutine to replace $\mathcal{A}_\Theta$ by taking advantage of quantum linear system algorithms (Harrow et al., 2009; Childs et al., 2017). This latter approach is motivated by the work of Nannicini (2024), among others, that achieves a speedup under some assumptions for the simplex method by using such quantum subroutines.

## 7.2 Quantum-classical resolution

Let us show how we can provide a quadratic speedup over the classical complexity by using the quantum algorithm of Dürr and Høyer (1996) straightforwardly. This algorithm has been introduced in Definition 5.2.6 for a high-level description, and in Definition 6.1.1 concerning the circuit description that we use below.

Let $U_\Theta$, respectively $U_P$, the quantum circuit implementing $\mathcal{A}_\Theta$, respectively $\mathcal{A}_P$ (see Observation 2.3.1). More precisely, we refer to $\mathcal{A}_\Theta$ as the algorithm that, given the input $L \subseteq \{1,\dots,2n+s\}$ such that $|L| = s$, outputs the vector

$$v_\theta := \left(\sum_{k\in[s]} a_{ki}\theta_k\right)_{i\in[n]},$$

and the value

$$v^0 := \sum_{k\in[s]} r_k\theta_k\,,$$

where $\theta$ is the unique solution of $(S_L)$ if the latter subsystem is of rank $s$ and if its unique solution is non-negative, and $(+\infty)^s$ otherwise. Namely, for $L$ such as $(S_L)$ admits a unique non-negative solution,

$$U_\Theta\,|a,b,c,d,L\rangle\,|0\rangle = |a,b,c,d,L\rangle\,\left|v_\theta, v^0\right\rangle\,,$$

and

$$U_P\left|a,b,c,d,L,v_\theta,v^0\right\rangle|0\rangle = \left|a,b,c,d,L,v_\theta,v^0\right\rangle|z^*_\theta\rangle\,,$$

up to a normalization factor. Thus, the application of $\mathrm{QMF}[U_Q U_\Theta]$ to the register that superposes all the $\binom{2n+s}{s}$ possible sets of indexes of rows outputs $u^*$. Specifically,

$$\mathrm{QMF}[U_P U_\Theta] \sum_{\substack{L\subseteq\{1,\dots,2n+s\}\\|L|=s}} \frac{1}{\sqrt{s}} |a,b,c,d,L\rangle |0\rangle |0\rangle |0\rangle =$$

$$\sum_{\substack{L\subseteq\{1,\dots,2n+s\}\\|L|=s}} \frac{1}{\sqrt{s}} |a,b,c,d,L\rangle |v_\theta\rangle \left|\arg\min_{\theta\in\Theta} z^*_\theta\right\rangle \left|\underbrace{\min_{\theta\in\Theta} z^*_\theta}_{u^*}\right\rangle .$$

Notice that we superpose all $L$ of cardinality $s$ to ease the superposition in practice. Consequently, we define the action of $U_\Theta$ on $L$ such that $(S_L)$ does not admit a unique non-negative solution to return the vector $(+\infty)^s$. It follows that $(v_\theta, v^0)$ will be set to $(+\infty)^{n+1}$ and will not change the solution of the minimization problem (7.2).

**Theorem 7.2.1.** *The complexity to solve* (MIN-MAX) *with high probability, using the Quantum Minimum Finding as a quantum routine, is* $\mathcal{O}(n^{\frac{s}{2}}(g(s,n)+f(n)))$.

*Proof.* According to Observation 2.3.1 and Observation 2.2.12, the complexity of $U_P U_\Theta$ is $\mathcal{O}(g(s,n)+f(n))$. Thus, applying Quantum Minimum Finding to this operator provides a quadratic speedup over the search of the space whose size is $\mathcal{O}(n^s)$ (see Definition 5.2.6 and Definition 6.1.1). Consequently, the hybrid algorithm's complexity is $\mathcal{O}(n^{\frac{s}{2}}(g(s,n)+f(n)))$. □

We showed in this section that Quantum Minimum Finding reduces the complexity associated with finding the best solution among all problems $(P_\theta)$ which are solved classically. In some cases, it might be advantageous to use an additional quantum subroutine in $\mathcal{A}_\Theta$ that could improve its complexity $g(s,n)$. We discuss in the next section the proposed additional quantum subroutine. Before that, we state below two necessary conditions to make relevant the search for such a quantum subroutine.

**Remark 7.2.2.** *The reduction of the complexity $g(s,n)$ is relevant if the number of inequalities describing the polytope $\Xi$ depends on $n$.*

For instance, robust optimization with locally budgeted uncertainty (Goerigk and Lendl, 2021) considers an uncertainty set polytope with $s=\mathcal{O}(n)$ inequalities. It amounts to partitioning the uncertainty parameters into *regions*, and applying to each region a budgeted uncertainty set.

**Remark 7.2.3.** *In addition to the remark above, the reduction of the complexity $g(n)$ is relevant if the complexity of solving the nominal problem* (7.1)*, and thus problem $(P_\theta)$, is smaller than $g(n)$. In other words, $f(n)$ must be dominated by the classical complexity $g(n)$ so that $g(n)$ dominates the term $(g(n)+f(n))$ (see complexity in Theorem 7.2.1).*

For example, this condition is satisfied for nominal problems such as the Spanning Tree problem or the Shortest Path problem if we consider $s=\mathcal{O}(n)$, leading to $g(n)=\mathcal{O}(n^3)$ with classical Gaussian elimination.

## 7.3 Additional quantum subroutine for linear systems

Let us begin with some preliminaries that we use for the description of the additional quantum subroutine.

### 7.3.1 Preliminaries

We note $\|x\|_2$ the Euclidean norm of a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, specifically,

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}\,.$$

We note $\|M\|_*$ the spectral norm of a matrix $M$,

$$\|M\|_* = \max_i \sigma_i\,,$$

where $\sigma_i$ is a singular value of $M$, namely the square root of an eigenvalue of $MM^\dagger$. Consequently, for $M$ symmetric, the singular values are the absolute values of its eigenvalues. Eventually, we note $\|M\|_F$ the Frobenius norm of matrix $M = (m_{ij}) \in \mathbb{R}^{n\times m}$ defined as follows:

$$\|M\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} m_{ij}^2}\,.$$

**Definition 7.3.1** (Amplitude encoding)**.** *For a given vector $x \in \mathbb{R}^{2^n}$, we refer to its amplitude encoding as*

$$|x\rangle = \sum_{i\in\{0,1\}^n} \frac{x_i}{\|x\|_2}\,|i\rangle\ .$$

Henceforth, we designate by $P_M$, respectively $P_x$, the unitary operators representing the oracles of matrix $M$, respectively vector $x$. In other words, $P_M$ refers to the procedure that computes the locations and values of the nonzero entries of $M$, and $P_x$ refers to the procedure that produces the state $|x\rangle$.

**Proposition 7.3.2** (Quantum Linear System Algorithm (Childs et al., 2017))**.** *Let us consider the linear system*

$$Gx = h\,,$$

*where $h \in \mathbb{C}^N$, and $G \in \mathbb{C}^{N\times N}$ is an Hermitian invertible matrix such that $\|G\|_* \leq 1$ and $G$ has a known condition number $\kappa$. Let $\epsilon > 0$ be a precision parameter. The Quantum Linear System Algorithm (QLSA) of Childs et al. (2017) produces a quantum state $|\tilde{x}\rangle$ such that*

$$\left\| |G^{-1}h\rangle - |\tilde{x}\rangle \right\|_2 \leq \epsilon\,.$$

*Henceforth, we use $\tilde{\mathcal{O}}$ which is the usual asymptotic notation that ignores the polylogarithmic factors in the complexity. This algorithm requires $\mathcal{O}(d\kappa \cdot polylog(\frac{d\kappa}{\epsilon})) = \tilde{\mathcal{O}}(d\kappa)$ queries to $P_G$ and $P_h$, where $d$ is the sparsity of $G$. Observe that the dependence on $\epsilon$ is polylogarithmic.*

Notice that the Hermitian condition, which is equivalent to considering symmetric matrices because we work on real matrices, can be easily relaxed as explained by Harrow et al. (2009). The spectral norm condition can also be relaxed by re-scaling the coefficients of $G$ if we know an upper bound $\alpha_G$ of the largest absolute value of its eigenvalues. Indeed, the spectral theorem ensures that $G = PDP^{-1}$, where $P$ is orthonormal and $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix of real coefficients. Thus, $G' := \frac{1}{\alpha_G} \cdot G$, where $\alpha_G \geq \max_{i\in[N]} |\lambda_i|$, satisfies the spectral norm condition $\|G'\|_* \leq 1$. Notice that, hereafter, we consider symmetric matrices such that $\|G\|_* = 1$, namely, we assume that we can estimate their maximum singular value $\max_{i\in[N]} |\lambda_i|$.

This will be helpful to compute an upper bound of the error produced by our algorithm in the next subsection.

**Proposition 7.3.3** (SWAP test)**.** *Let $|\psi\rangle$ and $|\phi\rangle$ be two quantum states. The SWAP test is a quantum circuit that prepares a qubit for which the probability of measuring 1 is equal to*

$$p = \frac{1}{2} - \frac{1}{2}\left|\langle\psi|\phi\rangle\right|^2 .$$

*In other words, the SWAP test prepares a qubit that represents a Bernoulli random variable $\mathcal{B}(p)$ where the probability of success is $p$.*

*Let $(X_1, \dots, X_N)$ be $N$ Bernoulli random variables $\mathcal{B}(p)$ for an unknown parameter $p \in [0,1]$. Let $Y = \frac{1}{N}\sum_{i=1}^{N} X_i$ be the resulting mean random variable. Thus, $Y := \hat{p}(N)$ is the estimation of the unknown parameter $p$. The Chebytchev inequality gives, for $\epsilon > 0$,*

$$\mathbb{P}(|Y - \mathbb{E}(Y)| \geq \epsilon) \leq \frac{\mathbb{V}(Y)}{\epsilon^2} .$$

*We have $\mathbb{E}(Y) = p$ and $\mathbb{V}(Y) = \frac{p(1-p)}{N} \leq \frac{1}{4N}$. The Chebytchev inequality results in*

$$\mathbb{P}(|\hat{p}(N) - p| \geq \epsilon) \leq \frac{1}{4N\epsilon^2} .$$

*Thus, making $N = \mathcal{O}(\frac{1}{\epsilon^2})$ samples ensures the additive error of the estimated parameter to be at most $\epsilon$. Specifically, it computes $\hat{p}(N)$ such that $|\hat{p}(N) - p| \leq \epsilon$ with high probability.*

**Proposition 7.3.4** (Variable Time Quantum Algorithm to estimate the norm (Chakraborty et al., 2018))**.** *Let us consider the linear system of Proposition 7.3.2 under the same assumptions. The variable time quantum algorithm of Chakraborty et al. (2018) outputs the estimation of the norm of the solution, with a relative error $\epsilon$, with $\tilde{\mathcal{O}}(\frac{1}{\epsilon} d\kappa)$ calls to $P_G$ and $P_h$. Specifically, the algorithm outputs $\tilde{n}$ such that*

$$\left|\tilde{n} - \left\|G^{-1}h\right\|_2\right| \leq \epsilon \left\|G^{-1}h\right\|_2 .$$

### 7.3.2 Quantum Linear System Algorithm subroutine

Under specific considerations on the polytope $\Xi$ and assuming access to QRAM, one can hope to reduce the complexity $g(s,n)$ by solving linear systems with the Quantum Linear System Algorithms (QLSA) of Harrow et al. (2009), or a latest version such as the one of Childs et al. (2017). Indeed, these algorithms provide, under some assumptions, an exponential speedup to the problem of finding a summary statistic of the solution of a linear system. In our case, it is relevant to consider QLSA to speed up the resolution of linear systems because the dependency of $(P_\theta)$ on $\theta \in \Theta$ amounts to the scalars $\sum_{k\in[s]} a_{ki}\theta_k$ for $i \in [n]$ and $\sum_{k\in[s]} r_k\theta_k$. In other words, to express $(P_\theta)$, we only need to compute the $n$ scalars composing the vector $v_\theta = (\sum_{k\in[s]} a_{ki}\theta_k)_{i\in[n]}$ plus the scalar $v^0 = \sum_{k\in[s]} r_k\theta_k$, and do not need to know the value of each coordinate of $\theta$.

In what follows, we propose a quantum counterpart $\mathcal{A}^q_\Theta$, using QLSA, to the previous classical algorithm $\mathcal{A}_\Theta$, and analyze the resulting complexity. Let us consider the subsystem

$$G(L)\theta = h(L) \qquad (S_L)$$

associated with $L \subseteq \{1, \dots, 2n+s\}$ and $|L| = s$ such that it admits a unique non-negative solution $\theta$ (unknown so far). As mentioned above, to get the expression of the function $g_\theta$, we need to

compute the $n$ components of vector $v_\theta$,

$$v_{\theta,i} = \sum_{k\in[s]} a_{ki}\theta_k = \|\theta\|_2 \cdot \|a_i\|_2 \cdot \langle a_i|\theta\rangle \ , \quad \forall i \in [n] \, ,$$

and the value

$$v^0 = \sum_{k\in[s]} r_k\theta_k = \|\theta\|_2 \cdot \|r\|_2 \cdot \langle r|\theta\rangle \ .$$

For $\mathcal{A}^q_\Theta$ to be correct, we make the following assumptions. First, for any $L$ of interest, we suppose that the conditions of Proposition 7.3.2 are satisfied. Specifically, we suppose that $\|G(L)\|_* = 1$, and that its condition number $\kappa(L)$ is known (or at least an upper bound). We note $d(L)$ the sparsity of $G(L)$. Second, we assume that $A$ and $r$ have non-negative coefficients. In that case, the expression of the vector $\beta$ in the objective function of $(P_\theta)$ simplifies in, $\forall i \in [n]$,

$$\beta_{\theta,i} = \left[1 - \sum_{k\in[s]} a_{ki}\theta_k\right]^+ \ .$$

The quantum routine constituting $\mathcal{A}^q_\Theta$ is described in Algorithm 12, where the estimation of the scalar product by Algorithm 11 represents the main subroutine. We begin by presenting the latter, for a generic non-negative $s$-vector $b$.

**Algorithm 11:** Estimation of scalar $\langle b|\theta\rangle$

**Input:** System $(S_L)$, $b \in \mathbb{R}^s_+$, precision parameters $\epsilon_1, \epsilon_2 > 0$
**Output:** Estimation of $\langle b|\theta\rangle$
**repeat**
  Prepare $\left|\tilde{\theta}\right\rangle$ with QLSA (Proposition 7.3.2) with precision $\epsilon_1$;
  Apply the SWAP test (Proposition 7.3.3) to $|b\rangle$ and $\left|\tilde{\theta}\right\rangle$, and measure the resulting qubit;
**until** $\mathcal{O}(\epsilon_2^{-2})$ *times*;
Compute the estimated probability of success $\hat{p}$;
Return $\sqrt{1-2\hat{p}}$;

Notice that Algorithm 11 applies to a vector $b$ with non-negative coefficients so that $\langle b|\theta\rangle \geq 0$ (because $\theta$ is non-negative), thus ensuring that $\langle b|\theta\rangle = \sqrt{1-2p}$ for $p = \frac{1}{2} - \frac{1}{2}\left|\langle b|\theta\rangle\right|^2$.

It results from Algorithm 11 the algorithm $\mathcal{A}^q_\Theta$, described in Algorithm 12, as follows.

**Theorem 7.3.5.** *Under the assumption to have access to a QRAM, implying that a query to oracles $P_{G(L)}$ and $P_{h(L)}$ is at most polylogarithmic, the complexity of Algorithm 12 is*

$$g(s,n) = \tilde{\mathcal{O}}((\epsilon_3^{-1} + \epsilon_2^{-2}n + \epsilon_5^{-2}) \max_L(\kappa(L)d(L))) \, .$$

*Proof.* Let us detail the complexity of each step of Algorithm 12.

- According to Proposition 7.3.4, the complexity of Step 1 is $\tilde{\mathcal{O}}(\epsilon_3^{-1}\kappa(L)d(L))$ queries to $P_{G(L)}$ and $P_{h(L)}$.

**Algorithm 12:** Estimation of vector $v_\theta$ and scalar $v^0$ (Algorithm $\mathcal{A}^q_\Theta$)

**Input:** System $(S_L)$, precision parameters $\epsilon_1, \epsilon_2, \epsilon_3 > 0$
**Output:** Estimation of vector $v_\theta$ and scalar $v^0$

1 Estimate $\|\theta\|_2$ with the Variable Time Quantum Algorithm (Proposition 7.3.4) with precision $\epsilon_3$. Note $\tilde{w}$ the output;
**for** $i \in [n]$ **do**
2 Estimate $\langle a_i|\theta\rangle$ with Algorithm 11 with precision $\epsilon_1, \epsilon_2$. Note $\sqrt{\tilde{e}}$ the output;
3 Compute and return $\|a_i\|_2 \tilde{w}\sqrt{\tilde{e}}$ (this is an estimation of $v_{\theta,i}$)
4 Estimate $\langle r|\theta\rangle$ with Algorithm 11 with precision $\epsilon_4, \epsilon_5$. Note $\sqrt{\tilde{e}^0}$ the output;
5 Compute and return $\|r\|_2 \tilde{w}\sqrt{\tilde{e}^0}$ (this is an estimation of $v^0$)

- According to Proposition 7.3.3, the complexity of Step 2 is $\mathcal{O}(\epsilon_2^{-2}C)$, where $C$ is the complexity of preparing $|a_i\rangle$ and $\left|\tilde{\theta}\right\rangle$ for a given $i \in [n]$. According to Proposition 7.3.2, the latter complexity is $\tilde{\mathcal{O}}(\kappa(L)d(L))$ queries to $P_{G(L)}$ and $P_{h(L)}$. Thus, the overall complexity is $\tilde{\mathcal{O}}(\epsilon_2^{-2}\kappa(L)d(L))$ queries to $P_{G(L)}$ and $P_{h(L)}$, and $\mathcal{O}(\epsilon_2^{-2})$ queries to $P_{a_i}$.
- Consequently, the complexity of the loop is $\tilde{\mathcal{O}}(\epsilon_2^{-2}n\kappa(L)d(L))$ queries to $P_{G(L)}$ and $P_{h(L)}$, and $\mathcal{O}(\epsilon_2^{-2})$ queries to each oracle representing a column of $A$.
- With the same justification, the complexity of Step 4 is $\tilde{\mathcal{O}}(\epsilon_5^{-2}\kappa(L)d(L))$ queries to $P_{G(L)}$ and $P_{h(L)}$, and $\mathcal{O}(\epsilon_5^{-2})$ queries to $P_r$.

□

Let us suppose that $\kappa(L)$ and $d(L)$ are polylogarithmic in $n$, which is a common assumption for QLSA to provide a potential speedup. Thus, the algorithm $\mathcal{A}^q_\Theta$ provides a speedup over $\mathcal{A}_\Theta$ (with Gaussian elimination) if $\epsilon_2 = \mathcal{O}(\frac{1}{n^\alpha})$ for $\alpha < 1$. Next, we compute an upper bound of the error induced by Algorithm 12 on the optimal solution of (MIN-MAX). We show that this upper bound, either for the relative or absolute error, is unbounded for such $\epsilon_2$. This leads us to believe that, unless the upper bound is too loose, the use of such a quantum subroutine does not provide any speedup in our case.

**Proposition 7.3.6.** *The error between the optimal value $u^*$ of* (MIN-MAX) *and the output $\tilde{u}^*$ of its resolution with the quantum subroutine $\mathcal{A}^q_\Theta$ has the following upper bound:*

$$|\tilde{u}^* - u^*| \leq s \max_L \kappa(L) \left( \epsilon \sum_i \|a_i\|_2 + \epsilon^0 \|r\|_2 \right),$$

*where*

$$\epsilon = (1+\epsilon_3)\left(\sqrt{\epsilon_2} + \epsilon_1 \cdot \sum_k \frac{a_{ki}}{\|a_i\|_2}\right) + \epsilon_3$$

*and*

$$\epsilon^0 = (1+\epsilon_3)\left(\sqrt{\epsilon_5} + \epsilon_4 \cdot \sum_k \frac{r_k}{\|r\|_2}\right) + \epsilon_3 .$$

*Proof.* We sketch the main steps of the proof, differing the details to the end of this chapter (Section 7.5). To begin with, we list the errors coming from the different steps of Algorithm 12. Regarding Step 1, the Variational Time Quantum Algorithm outputs $\tilde{w} := \tilde{v} \left\| h(L) \right\|_2$ such that

$$\left| \frac{\tilde{v} - \left\| G(L)^{-1} \left| h(L) \right\rangle \right\|_2}{\left\| G(L)^{-1} \left| h(L) \right\rangle \right\|_2} \right| \leq \epsilon_3 \,,$$

namely, such that

$$|\tilde{w} - \left\| \theta \right\|_2| \leq \epsilon_3 \cdot \left\| \theta \right\|_2 \,.$$

Regarding Step 2, Algorithm 11 outputs $\sqrt{\tilde{e}}$ such that

$$\left| \tilde{e} - \left\langle a_i \middle| \tilde{\theta} \right\rangle^2 \right| \leq \epsilon_2 \,,$$

where the quantum state $\left| \tilde{\theta} \right\rangle$ prepared by QLSA satisfies

$$\left\| |\tilde{\theta}\rangle - \left| G(L)^{-1} h(L) \right\rangle \right\|_2 \leq \epsilon_1 \,.$$

Similarly, regarding Step 4, Algorithm 11 outputs $\sqrt{\tilde{e}^0}$ such that

$$\left| \tilde{e} - \left\langle r \middle| \tilde{\theta} \right\rangle^2 \right| \leq \epsilon_5 \,,$$

where the quantum state $\left| \tilde{\theta} \right\rangle$ is prepared by QLSA such that

$$\left\| |\tilde{\theta}\rangle - \left| G(L)^{-1} h(L) \right\rangle \right\|_2 \leq \epsilon_4 \,.$$

Next, we compute for a given $i \in [n]$ an upper bound of the error of the value resulting in Step 3 that is meant to approximate $v_{\theta,i}$. Notice that according to the definition of the amplitude encoding (see Definition 7.3.1) and the inner product for real vectors, $v_{\theta,i} = \left\| \theta \right\|_2 \left\| a_i \right\|_2 \langle a_i | \theta \rangle$ . We define

$$\Delta^i = \underbrace{\tilde{v} \cdot \left\| h(L) \right\|_2 \cdot \left\| a_i \right\|_2 \sqrt{\tilde{e}}}_{\text{estimation of } v_{\theta,i}} - \left\| \theta \right\|_2 \left\| a_i \right\|_2 \langle a_i | \theta \rangle \,,$$

and can show that

$$|\Delta^i| \leq \epsilon \sqrt{s} \kappa(L) \left\| a_i \right\|_2 \left\| h(L) \right\|_2 \,,$$

where

$$\epsilon = (1 + \epsilon_3) \left( \sqrt{\epsilon_2} + \epsilon_1 \cdot \sum_k \frac{a_{ki}}{\left\| a_i \right\|_2} \right) + \epsilon_3 \,.$$

This leads to

$$\sum_{i \in [n]} |\Delta^i| \leq \epsilon \sqrt{s} \kappa(L) \left\| h(L) \right\|_2 \sum_{i \in [n]} \left\| a_i \right\|_2 \,.$$

Similarly, we find the following upper bound of the error resulting in Step 5 of Algorithm 12 that is meant to approximate $\sum_{k \in [s]} r_k \theta_k$.

$$\left| \sum_{k \in [s]} r_k \tilde{\theta}_k - \sum_{k \in [s]} r_k \theta_k \right| \leq \epsilon^0 \sqrt{s} \kappa(L) \left\| r \right\|_2 \left\| h(L) \right\|_2 \,,$$

where

$$\epsilon^0 = (1+\epsilon_3)\left(\sqrt{\epsilon_5} + \epsilon_4 \cdot \sum_{k\in[s]} \frac{r_k}{\|r\|_2}\right) + \epsilon_3\,.$$

Thus, for any $\theta$ corresponding to the unique non-negative solution of a system $(S_L)$ and $\tilde{\theta}$ the approximate solution provided by Algorithm 12, and for any $y \in \mathcal{Y}$, we have

$$\left|g_{\tilde{\theta}}(y) - g_\theta(y)\right| \leq \sum_{i\in[sn]} \left|\Delta^i\right| + \left|\sum_{k\in[s]} r_k(\tilde{\theta}_k - \theta_k)\right|\,.$$

This results in the stated upper bound of $|\tilde{u}^* - u^*|$. □

Previously, we deduced from Theorem 7.3.5 that our hybrid algorithm with additional quantum subroutine $\mathcal{A}^q_\Theta$ could achieve a speedup for a precision parameter $\epsilon_2 = \mathcal{O}(\frac{1}{n^\alpha})$ for $\alpha < 1$. However, Proposition 7.3.6 shows that the error induced by $\mathcal{A}^q_\Theta$ cannot be *controlled* and reduced to zero for such an $\epsilon_2$ (even if we try to consider a relative error). This limitation seems to come from the fact that in Step 3 and Step 5, QLSA prepares a quantum state encoding the normalized solution of a linear system for which the extraction of summary statistic $(v_\theta, v^0)$ costs too much and neutralizes the exponential speedup of QLSA. Thus, the use of QLSA to get numerical values seems to be a bottleneck in our case.

## 7.4 Conclusion

In this chapter, we investigated the use of quantum subroutines to tackle robust combinatorial optimization problems with polyhedral uncertainty sets. More precisely, we considered the Quantum Minimum Finding, which achieves a quadratic speedup, and the use of a Quantum Linear System Algorithm, which does not seem to provide any speedup for these problems due to the numerical output we ask for. Further investigations on this subject are considered for future work. One is to see if the upper bound of the overall error of Proposition 7.3.6 is tight or not. We believe it may not be so and hope that a tighter bound could reduce the error and make possible a quantum speedup. Another question is: what do the assumptions we have made about the subsystems $(S_L)$ (symmetric and spectral norm conditions), plus the positivity of the coefficients of $A$ and $r$, imply on the structure of the polytope $\Xi$ ?

## 7.5 Details on the error computation

In this subsection, we detail the proof of the following proposition that provides an upper bound of the error induced by the quantum algorithm $A^q_\Theta$ (Algorithm 12).

**Proposition 7.3.6.** *The error between the optimal value $u^*$ of* (MIN-MAX) *and the output $\tilde{u}^*$ of its resolution with the quantum subroutine $\mathcal{A}^q_\Theta$ has the following upper bound:*

$$|\tilde{u}^* - u^*| \leq s \max_L \kappa(L)\left(\epsilon \sum_i \|a_i\|_2 + \epsilon^0 \|r\|_2\right)\,,$$

*where*

$$\epsilon = (1+\epsilon_3)\left(\sqrt{\epsilon_2} + \epsilon_1 \cdot \sum_k \frac{a_{ki}}{\|a_i\|_2}\right) + \epsilon_3$$

*and*

$$\epsilon^0 = (1+\epsilon_3)\left(\sqrt{\epsilon_5} + \epsilon_4 \cdot \sum_k \frac{r_k}{\|r\|_2}\right) + \epsilon_3 \,.$$

*Proof.* Firstly, let us list the errors coming from the different steps of Algorithm 12. Regarding Step 1, the Variational Time Quantum Algorithm outputs $\tilde{w} := \tilde{v}\,\|h(L)\|_2$ such that

$$\left|\frac{\tilde{v} - \left\|G(L)^{-1}\,|h(L)\rangle\right\|_2}{\|G(L)^{-1}\,|h(L)\rangle\|_2}\right| \leq \epsilon_3 \,,$$

namely, such that

$$|\tilde{w} - \|\theta\|_2| \leq \epsilon_3 \cdot \|\theta\|_2 \,.$$

Regarding Step 2, Algorithm 11 outputs $\sqrt{\tilde{e}}$ such that

$$\left|\tilde{e} - \left\langle a_i \middle| \tilde{\theta} \right\rangle^2\right| \leq \epsilon_2 \,,$$

where the quantum state $\left|\tilde{\theta}\right\rangle$ prepared by QLSA satisfies

$$\left\| |\tilde{\theta}\rangle - \left|G(L)^{-1}h(L)\right\rangle \right\|_2 \leq \epsilon_1 \,.$$

Similarly, regarding Step 4, Algorithm 11 outputs $\sqrt{\tilde{e}^0}$ such that

$$\left|\tilde{e} - \left\langle r \middle| \tilde{\theta} \right\rangle^2\right| \leq \epsilon_5 \,,$$

where the quantum state $\left|\tilde{\theta}\right\rangle$ is prepared by QLSA such that

$$\left\| |\tilde{\theta}\rangle - \left|G(L)^{-1}h(L)\right\rangle \right\|_2 \leq \epsilon_4 \,.$$

Next, we compute for a given $i \in [n]$ an upper bound of the error of the value resulting in Step 3 that is meant to approximate $v_{\theta,i}$. Notice that according to the definition of the amplitude encoding (see Definition 7.3.1) and the dot product for real vectors, $v_{\theta,i} = \|\theta\|_2 \|a_i\|_2 \langle a_i|\theta\rangle$. Let us compute

$$\Delta^i = \underbrace{\tilde{v} \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sqrt{\tilde{e}}}_{\text{estimation of } \|\theta\|_2\|a_i\|_2\langle a_i|\theta\rangle} - \|\theta\|_2 \|a_i\|_2 \langle a_i|\theta\rangle \,.$$

According to triangular inequality,

$$|\Delta^i| \leq |\Delta^i_{\text{swap}}| + |\Delta^i_{\text{sol}}| + |\Delta^i_{\text{norm}}|$$

with

$$\Delta^i_{\text{swap}} = \tilde{v} \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sqrt{\tilde{e}} - \tilde{v} \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sqrt{\left\langle a_i \middle| \tilde{\theta} \right\rangle^2} \,,$$

$$\Delta^i_{\text{sol}} = \tilde{v} \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \left\langle a_i \middle| \tilde{\theta} \right\rangle - \tilde{v} \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \langle a_i|\theta\rangle \,,$$

$$\Delta^i_{\text{norm}} = \tilde{v} \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \langle a_i|\theta\rangle - \|\theta\|_2 \cdot \|a_i\|_2 \langle a_i|\theta\rangle \,.$$

First, let us find an upper bound on $\left|\Delta^i_{\text{swap}}\right|$.

$$\begin{aligned}
\left|\Delta^i_{\text{swap}}\right| &\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \left|\sqrt{\tilde{e}} - \sqrt{\left\langle a_i \middle| \tilde{\theta} \right\rangle^2}\right| \\
&\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sqrt{\left|\tilde{e} - \left\langle a_i \middle| \tilde{\theta} \right\rangle^2\right|} \quad \text{(because square root function is Hölder 1/2)} \\
&\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sqrt{\epsilon_2} \quad \text{(according to Step 2)}\,.
\end{aligned}$$

Second, we can find the following upper bound on $|\Delta^i_{\text{sol}}|$.

$$\begin{aligned}
|\Delta^i_{\text{sol}}| &\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \left|\left\langle a_i \middle| \tilde{\theta} \right\rangle - \langle a_i | \theta \rangle\right| \\
&\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \left|\sum_k \frac{a_{ki}}{\|a_i\|_2} \left(\frac{\tilde{\theta}_k}{\left\|\tilde{\theta}\right\|_2} - \frac{\theta_k}{\|\theta\|_2}\right)\right| \\
&\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sum_k \frac{a_{ki}}{\|a_i\|_2} \left|\frac{\tilde{\theta}_k}{\left\|\tilde{\theta}\right\|_2} - \frac{\theta_k}{\|\theta\|_2}\right| \\
&\leq |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 \sum_k \frac{a_{ki}}{\|a_i\|_2} \epsilon_1 \quad \text{(according to Step 2)}\,.
\end{aligned}$$

Lastly, an upper bound on $|\Delta^i_{\text{norm}}|$ is found.

$$\begin{aligned}
|\Delta^i_{\text{norm}}| &= \|a_i\|_2 \langle a_i | \theta \rangle \left|\tilde{v} \|h(L)\|_2 - \|\theta\|_2\right| \\
&\leq \|a_i\|_2 \langle a_i | \theta \rangle \|\theta\|_2 \epsilon_3 \quad \text{(according to Step 1)} \\
&\leq \|a_i\|_2 \|\theta\|_2 \epsilon_3 \quad \text{(because } \langle a_i | \theta \rangle \in [0,1]\text{)}\,.
\end{aligned}$$

According to each upper bounds above-mentioned, it results an upper bound on $|\Delta^i|$.

$$\begin{aligned}
|\Delta^i| &\leq \left(\sqrt{\epsilon_2} + \sum_k \frac{a_{ki}}{\|a_i\|_2} \epsilon_1\right) \cdot |\tilde{v}| \cdot \|h(L)\|_2 \cdot \|a_i\|_2 + \epsilon_3 \cdot \|a_i\|_2 \|\theta\|_2 \\
&\leq (1+\epsilon_3) \left(\sqrt{\epsilon_2} + \sum_k \frac{a_{ki}}{\|a_i\|_2} \epsilon_1\right) \cdot \|a_i\|_2 \|\theta\|_2 + \epsilon_3 \cdot \|a_i\|_2 \|\theta\|_2 \quad \text{(according to Step 1)} \\
&\leq \left[(1+\epsilon_3) \left(\sqrt{\epsilon_2} + \sum_k \frac{a_{ki}}{\|a_i\|_2} \epsilon_1\right) \; + \epsilon_3\right] \cdot \|a_i\|_2 \|\theta\|_2 \,.
\end{aligned}$$

Next, we remove the dependency on $\theta$ of the upper bound by replacing it with values regarding the initial problem, namely $G(L)$ and $h(L)$.

$$\|\theta\|_2 = \left\|G(L)^{-1} h(L)\right\|_2 \leq \left\|G(L)^{-1}\right\|_F \cdot \|h(L)\|_2 \,,$$

and

$$\left\|G(L)^{-1}\right\|_F \leq \sqrt{r} \left\|G(L)^{-1}\right\|_* \,,$$

where $r$ is the rank of $G(L)^{-1}$, equal to $s$ here. Moreover,

$$\left\|G(L)^{-1}\right\|_* = \sigma_{\max}(G(L)^{-1}) = \frac{1}{\sigma_{\min}(G(L))}$$

where $\sigma_{\min}(G(L)) = \frac{1}{\kappa(L)}$ because we assumed that $\|G(L)\|_* = 1$. Thus,

$$\|\theta\|_2 \leq \sqrt{s}\kappa(L)\,\|h(L)\|_2\ .$$

Eventually,

$$|\Delta^i| \leq \epsilon\sqrt{s}\kappa(L)\,\|a_i\|_2\,\|h(L)\|_2\ ,$$

where

$$\epsilon = (1+\epsilon_3)\left(\sqrt{\epsilon_2} + \epsilon_1 \cdot \sum_k \frac{a_{ki}}{\|a_i\|_2}\right) + \epsilon_3\,,$$

leading to the following upper bound on the sum of all the errors of the loop in Algorithm 12:

$$\sum_i |\Delta^i| \leq \epsilon\sqrt{s}\kappa(L)\,\|h(L)\|_2 \sum_i \|a_i\|_2\ .$$

Similarly, we can show that the error resulting in Step 5 of Algorithm 12 is

$$\left|\sum_{k\in[s]} r_k\tilde{\theta}_k - \sum_{k\in[s]} r_k\theta_k\right| \leq \epsilon^0\sqrt{s}\kappa(L)\,\|r\|_2\,\|h(L)\|_2\ ,$$

where

$$\epsilon^0 = (1+\epsilon_3)\left(\sqrt{\epsilon_5} + \epsilon_4 \cdot \sum_k \frac{r_k}{\|r\|_2}\right) + \epsilon_3\,.$$

Thus, for any $\theta$ corresponding to the unique non-negative solution of a system $(S_L)$ and $\tilde{\theta}$ the approximate solution provided by Algorithm 12, and for any $y \in \mathcal{Y}$, we have

$$\begin{aligned}
\left|g_{\tilde{\theta}}(y) - g_\theta(y)\right| &= \left|\sum_i (\beta_{\tilde{\theta},i} - \beta_{\theta,i})y_i + \sum_k r_k(\tilde{\theta}_k - \theta_k)\right| \\
&\leq \sum_i \left|\Delta^i\right| + \left|\sum_k r_k(\tilde{\theta}_k - \theta_k)\right| \quad \text{(because } y_i \in \{0,1\} \text{ and the definition of } [.]^+\text{)} \\
&\leq \epsilon\sqrt{s}\kappa(L)\,\|h(L)\|_2 \sum_i \|a_i\|_2 + \epsilon^0\sqrt{s}\kappa(L)\,\|r\|_2\,\|h(L)\|_2 \\
&\leq \sqrt{s}\kappa(L)\,\|h(L)\|_2 \left(\epsilon\sum_i \|a_i\|_2 + \epsilon^0\,\|r\|_2\right)\ .
\end{aligned}$$

Consequently, we find an upper bound on the error between the optimal value $u^*$ of the nominal problem and the solution provided by the hybrid quantum-classical with the additional quantum subroutine $A^q_\Theta$.

$$\begin{aligned}
|\tilde{u}^* - u^*| &= \left|\min_{\theta\in\Theta} z^*_{\tilde{\theta}} - \min_{\theta\in\Theta} z^*_\theta\right| \\
&= \left|\min_{\theta\in\Theta}\min_{y\in\mathcal{Y}} g_{\tilde{\theta}}(y) - \min_{\theta\in\Theta}\min_{y\in\mathcal{Y}} g_\theta(y)\right| \\
&\leq \max_{\theta\in\Theta}\max_{y\in\mathcal{Y}} \left|g_{\tilde{\theta}}(y) - g_\theta(y)\right| \\
&\leq \sqrt{s}\max_L \kappa(L)\,\|h(L)\|_2 \left(\epsilon\sum_i \|a_i\|_2 + \epsilon^0\,\|r\|_2\right)\ .
\end{aligned}$$

By definition of $h(L)$, for any $L$, $\|h(L)\|_2 \leq \sqrt{\min(s,n)}$. For the locally budgeted uncertainty set, $s = \frac{n}{K}$ for a given $K \in \mathbb{N}^*$, leading to $\|h(L)\|_2 = \sqrt{s}$. Notice that for the general case $s = \mathcal{O}(n)$, the conclusion of how the precision parameters must scale with $n$ does not change.

To conclude,

$$|\tilde{u}^* - u^*| \leq s \max_L \kappa(L) \left( \epsilon \sum_i \|a_i\|_2 + \epsilon^0 \|r\|_2 \right)$$

where

$$\epsilon = (1+\epsilon_3) \left( \sqrt{\epsilon_2} + \epsilon_1 \cdot \sum_k \frac{a_{ki}}{\|a_i\|_2} \right) + \epsilon_3 \,,$$

and

$$\epsilon^0 = (1+\epsilon_3) \left( \sqrt{\epsilon_5} + \epsilon_4 \cdot \sum_k \frac{r_k}{\|r\|_2} \right) + \epsilon_3 \,.$$

□

# 8

# Conclusion and perspectives

This chapter concludes this thesis by summarizing the main contributions and providing some leads for future work on the continuation of what has been done so far. In our opinion, the most promising research directions arise in the continuation of the work on hybrid algorithms for scheduling problems as the study of Variational Quantum Algorithms seems restricted by the small size of possible numerical tests, and the work on robust optimization problems faces the limitation of the error computations.

## 8.1 Variational Quantum Algorithms and application to a railway timetabling problem

The first branch of quantum algorithms for solving combinatorial optimization problems we investigated is the branch of heuristics. Specifically, we tackled a large class of heuristics for gate-based quantum computers called Variational Quantum Algorithms (VQAs). Indeed, for several years, there has been a growing interest in VQAs, and more particularly in the Quantum Approximate Optimization Algorithm (QAOA), mostly because of their compatibility with NISQ computers. In Chapter 3, we provided a mathematical description of VQAs with a focus on QAOA, for which we dedicated special attention to the resolution of unconstrained binary problems with a polynomial objective function. We have described how VQAs raise interesting questions regarding the choice of guiding functions and quantum circuits, among others. A generic method to reformulate constrained problems with integer variables into unconstrained problems with binary variables has also been proposed to handle real-world problems with VQAs. We studied one of them in Chapter 4, which is a railway timetabling problem. For that, we simplified the SNCF nominal problems into two models: a Set Cover Problem and an Extended Bin Packing Problem. Solving such problems with QAOA depicted several trends in its performance, and highlighted the importance of the constrained problem reformulation.

Since the inception of such methods, researchers have argued about the potential advantages of QAOA over classical optimization algorithms (Farhi and Harrow, 2016; Lotshaw et al., 2021; Marwaha and Hadfield, 2022). The overview of the literature on this subject seems to indicate that there is currently no scientific evidence that QAOA will soon beat classical heuristics, as corroborated by our work on the railway timetabling problem. The available results even suggest the opposite, such as Hastings (2019) showing that local search algorithms can do better at low depth. Moreover, the current numerical results are hard to assess because they are run on small instances for which optimal or near-optimal solutions can be obtained easily with classical algorithms. Hopefully, the quickly growing capabilities of quantum computers will soon lead to a better understanding of the numerical efficiency of such algorithms for future research.

## 8.2 Hybrid algorithms for scheduling problems

The second branch of quantum algorithms investigated is the branch of exact algorithms. Contrary to the VQAs, they prove theoretical advantages but cannot be implemented today because of the huge amount of quantum resources they require. We presented in Chapters 5 and 6 a hybrid algorithm, based on the seminal idea of Ambainis et al. (2019) that combines Quantum Minimum Finding and dynamic programming, to address a broad class of NP-hard scheduling problems. Specifically, it tackles single-machine scheduling problems that satisfy a certain dynamic programming relying on the *addition* or the *composition* of optimal values of sub-instances. We also adapted this algorithm to tackle decision problems leading to the resolution of the 3-machine flowshop problem. This new method provides a quantum speed-up on the worst-case time complexity on the exponential part, sometimes at the cost of a pseudo-polynomial additional factor. Future work will be dedicated to widening the range of scheduling problems for which we can achieve a quantum speedup. This will necessitate finding NP-hard scheduling problems for which the description of a solution is a permutation, and that satisfy a dynamic programming property admitting a dichotomic version, relying either on an optimization or a decision problem. Consequently, the proposed algorithm should be adapted to this dynamic programming property. Besides adapting our algorithm to new problems, another path for future research is to find quantum brick(s) that could speed up exact exponential algorithms, as the Quantum Minimum Finding is used for dynamic programming. For instance, one could think of Grover Search (Grover, 1996) or Quantum Walks (Aharonov et al., 2001) for the exponential algorithms Sort-and-Search (Lenté et al., 2013) or Branch-and-Reduce (T'kindt et al., 2022) respectively.

More specifically, the first leads of research that we will explore in the future are the following. Regarding the adaptation of our algorithm to new scheduling problems, we will consider the 3-machine jobshop problem. This problem aims at scheduling a set of jobs, each consisting of multiple operations on 3 machines, aiming to minimize the total completion time. Contrary to the 3-machine flowshop problem, the order of the jobs is not fixed, thus a solution is not solely a permutation of the jobs. However, a promising description of a solution of the jobshop is the Bierwith vector (Bierwirth, 1995). This vector encodes a solution as a permutation with repetition of size $3n$, where $n$ is the number of jobs. Nevertheless, this is not trivial to see if its structure can satisfy a dynamic programming property with a dichotomic version, hence further investigation.

To explore the lead of combining a quantum subroutine with an exact exponential algorithm to achieve a speedup, one specific algorithm seems of particular interest. This is the Inclusion-Exclusion method, which addresses, among others, NP-hard scheduling problems (Ploton, 2023). This method consists in *relaxing* the job permutations describing a solution of the scheduling problem by admitting repetitions. Thus, the complexity of solving the problem with these relaxed solutions becomes easy, and the exponential part of the complexity *shifts* to counting the number of relaxed solutions in a specific set imposing some constraints. More precisely, only the knowledge of whether this set is empty or not is needed. Mathematically, this amounts to determine whether the quantity $S = \sum_{X \subseteq [n]} f(X)$, for which the computation of $f : 2^{[n]} \to \mathbb{R}^+$ is polynomial, is equal to 0 or not. For that, Inclusion-Exclusion computes $S$, which provides its complexity equal to $\mathcal{O}^*(2^n)$ (modulo pseudo-polynomial factors in some cases). This raises the following question: is it possible to know if $S \neq 0$ in a time smaller than $\mathcal{O}^*(2^n)$ ? This question is driven by the fact that, because only the knowledge of the strict positivity of $S$ is required, exact quantum algorithms could be of use, even with some error computations as depicted in the robust optimization case we tackled. We will first investigate the possibility of setting $S$ as the angle of a rotation quantum

gate. Then, if a rotation is detected by measuring (a limited number of times) the quantum state, it would mean that $S \neq 0$. This idea stems from the Quantum Linear System Algorithm of Harrow et al. (2009), where, after the eigendecomposition of the matrix of the linear system, the eigenvalues are encoded as angles of some rotation quantum gates.

## 8.3 Hybrid algorithm for robust optimization problems

Continuing on the branch of exact quantum algorithms, we studied robust optimization problems with polyhedral uncertainty sets. Starting from the classical algorithm of Omer et al. (2024) that solves such problems, we integrated two quantum subroutines. The first one, Quantum Minimum Finding (Dürr and Høyer, 1996), offers a polynomial speedup for the worst-case time complexity compared to its classical counterpart. The second one, the Quantum Linear System Algorithm of Childs et al. (2017), imposes specific assumptions under the linear systems to be solved, and thus on the uncertainty set of the robust problem at hand. Because of the introduction of errors through its use, this subroutine does not appear to achieve any speedup. However, this deserves further investigation, particularly to determine whether there are no tighter bounds on the error computations. Additionally, we will investigate the characterization of the uncertainty sets that satisfy the assumptions required by QLSA subroutine.

# Bibliography


Aharonov, D., Ambainis, A., Kempe, J., and Vazirani, U. (2001). Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59.

Alam, M., Ash-Saki, A., and Ghosh, S. (2020). Accelerating quantum approximate optimization algorithm using machine learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 686–689. IEEE.

Amaro, D., Modica, C., Rosenkranz, M., Fiorentini, M., Benedetti, M., and Lubasch, M. (2022). Filtering variational quantum algorithms for combinatorial optimization. *Quantum Science and Technology*, 7(1):015021.

Ambainis, A. (2007). Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239.

Ambainis, A., Balodis, K., Iraids, J., Kokainis, M., Prūsis, K., and Vihrovs, J. (2019). Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1783–1793. SIAM.

Barkoutsos, P. K., Nannicini, G., Robert, A., Tavernelli, I., and Woerner, S. (2020). Improving variational quantum optimization using CVaR. *Quantum*, 4:256.

Basso, J., Farhi, E., Marwaha, K., Villalonga, B., and Zhou, L. (2021). The quantum approximate optimization algorithm at high depth for maxcut on large-girth regular graphs and the sherrington-kirkpatrick model. *arXiv preprint arXiv:2110.14206*.

Bennett, C. H. (1973). Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532.

Bernstein, E. and Vazirani, U. (1993). Quantum complexity theory. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 11–20.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3):49–71.

Bickert, P., Grozea, C., Hans, R., Koch, M., Riehn, C., and Wolf, A. (2021). Optimising rolling stock planning including maintenance with constraint programming and quantum annealing. *arXiv preprint arXiv:2109.07212*.

Bierwirth, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, 17(2):87–92.

Blekos, K., Brand, D., Ceschini, A., Chou, C.-H., Li, R.-H., Pandya, K., and Summer, A. (2024). A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66.

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. (1998). Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505.

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

Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74.

Bravyi, S., Kliesch, A., Koenig, R., and Tang, E. (2020). Obstacles to variational quantum optimization from symmetry protection. *Physical Review Letters*, 125(26):260505.

Bravyi, S., Kliesch, A., Koenig, R., and Tang, E. (2022). Hybrid quantum-classical algorithms for approximate graph coloring. *Quantum*, 6:678.

Buchheim, C. and Kurtz, J. (2018). Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238.

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., et al. (2021). Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644.

Chakraborty, S., Gilyén, A., and Jeffery, S. (2018). The power of block-encoded matrix powers: improved regression techniques via faster hamiltonian simulation. *arXiv preprint arXiv:1804.01973*.

Chen, M. C., Goh, S. L., Sabar, N. R., Kendall, G., et al. (2021). A survey of university course timetabling problem: perspectives, trends and opportunities. *IEEE Access*, 9:106515–106529.

Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., and Spielman, D. A. (2003). Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 59–68.

Childs, A. M., Kothari, R., and Somma, R. D. (2017). Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950.

Coppersmith, D. (2002). An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*.

Crooks, G. E. (2018). Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv preprint arXiv:1811.08419*.

Cygan, M., Pilipczuk, M., Pilipczuk, M., and Wojtaszczyk, J. O. (2014). Scheduling partially ordered jobs faster than $2^n$. *Algorithmica*, 68:692–714.

Dalyac, C., Henriet, L., Jeandel, E., Lechner, W., Perdrix, S., Porcheron, M., and Veshchezerova, M. (2021). Qualifying quantum approaches for hard industrial optimization problems. a case study in the field of smart-charging of electric vehicles. *EPJ Quantum Technology*, 8(1):12.

de la Grand'rive, P. D. and Hullo, J.-F. (2019). Knapsack problem variants of qaoa for battery revenue optimisation. *arXiv preprint arXiv:1908.02210*.

De Palma, G., Marvian, M., Rouzé, C., and França, D. S. (2023). Limitations of variational quantum algorithms: a quantum optimal transport approach. *PRX Quantum*, 4(1):010309.

Dürr, C. and Høyer, P. (1996). A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*.

Egger, D. J., Marecek, J., and Woerner, S. (2021). Warm-starting quantum optimization. *Quantum*, 5:479.

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

Farhi, E., Goldstone, J., and Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

Farhi, E., Goldstone, J., Gutmann, S., and Sipser, M. (2000). Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*.

Farhi, E., Goldstone, J., Gutmann, S., and Zhou, L. (2022). The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size. *Quantum*, 6:759.

Farhi, E. and Harrow, A. W. (2016). Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*.

Fortran, I., Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). Numerical recipes. *Cambridge, UK, Cambridge University Press*.

Gilyén, A., Arunachalam, S., and Wiebe, N. (2019). Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. SIAM.

Gioda, I. (2021). *Seating Arrangement Optimization in COVID-19 Era: a quantum computing approach*. PhD thesis, Politecnico di Torino.

Giovannetti, V., Lloyd, S., and Maccone, L. (2008). Quantum random access memory. *Physical review letters*, 100(16):160501.

Glover, F., Kochenberger, G., Hennig, R., and Du, Y. (2022). Quantum bridge analytics i: a tutorial on formulating and using qubo models. *Annals of Operations Research*, 314(1):141–183.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145.

Goerigk, M. and Lendl, S. (2021). Robust combinatorial optimization with locally budgeted uncertainty. *Open Journal of Mathematical Optimization*, 2:1–18.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier.

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

Hadfield, S., Wang, Z., O'gorman, B., Rieffel, E. G., Venturelli, D., and Biswas, R. (2019). From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34.

Hall, L. A. (1998). Approximability of flow shop scheduling. *Mathematical Programming*, 82(1-2):175–190.

Harrow, A. W., Hassidim, A., and Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502.

Hastings, M. B. (2019). Classical and quantum bounded depth approximation algorithms. *arXiv preprint arXiv:1905.07047*.

Held, M. and Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162.

Hellemo, L., Barton, P. I., and Tomasgard, A. (2018). Decision-dependent probabilities in stochastic programs with recourse. *Computational Management Science*, 15(3):369–395.

Herman, D., Shaydulin, R., Sun, Y., Chakrabarti, S., Hu, S., Minssen, P., Rattew, A., Yalovetzky, R., and Pistoia, M. (2023). Constrained optimization via quantum zeno dynamics. *Communications Physics*, 6(1):219.

Herrman, R., Treffert, L., Ostrowski, J., Lotshaw, P. C., Humble, T. S., and Siopsis, G. (2021). Globally optimizing QAOA circuit depth for constrained optimization problems. *Algorithms*, 14(10):294.

Holmes, Z., Sharma, K., Cerezo, M., and Coles, P. J. (2022). Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1):010313.

Kadowaki, T. and Nishimori, H. (1998). Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355.

Kerenidis, I. and Prakash, A. (2020). A quantum interior point method for LPs and SDPs. *ACM Transactions on Quantum Computing*, 1(1):1–32.

Kerenidis, I., Prakash, A., and Szilágyi, D. (2021). Quantum algorithms for second-order cone programming and support vector machines. *Quantum*, 5:427.

Khumalo, M. T., Chieza, H. A., Prag, K., and Woolway, M. (2022). An investigation of IBM quantum computing device performance on combinatorial optimisation problems. *Neural Computing and Applications*, pages 1–16.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kitaev, A. Y. (1995). Quantum measurements and the Abelian stabilizer problem. *arXiv preprint quant-ph/9511026*.

Kurowski, K., Pecyna, T., Slysz, M., Różycki, R., Waligóra, G., and Weglarz, J. (2023). Application of quantum approximate optimization algorithm to job shop scheduling problem. *European Journal of Operational Research*.

Lawler, E. L. (1990). A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research*, 26:125–133.

Lenté, C., Liedloff, M., Soukhal, A., and T'kindt, V. (2013). On an extension of the sort & search method with application to scheduling theory. *Theoretical Computer Science*, 511:13–22.

Li, L., Fan, M., Coram, M., Riley, P., Leichenauer, S., et al. (2020). Quantum optimization with a novel Gibbs objective function and ansatz architecture search. *Physical Review Research*, 2(2):023074.

Lotshaw, P. C., Humble, T. S., Herrman, R., Ostrowski, J., and Siopsis, G. (2021). Empirical performance bounds for quantum approximate optimization. *Quantum Information Processing*, 20(12):1–32.

Lucas, A. (2014). Ising formulations of many NP problems. *Frontiers in Physics*, page 5.

Magniez, F., Santha, M., and Szegedy, M. (2007). Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424.

Marwaha, K. and Hadfield, S. (2022). Bounds on approximating Max $k$XOR with quantum and classical local algorithms. *Quantum*, 6:757.

Mastropietro, D., Korpas, G., Kungurtsev, V., and Marecek, J. (2023). Fleming-viot helps speed up variational quantum algorithms in the presence of barren plateaus. *arXiv preprint arXiv:2311.18090*.

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):1–6.

Miyamoto, M., Iwamura, M., Kise, K., and Gall, F. L. (2020). Quantum speedup for the minimum steiner tree problem. In *COCOON 2020, Atlanta, GA, USA, August 29–31, 2020, Proceedings*, pages 234–245. Springer.

Montanaro, A. (2015). Quantum walk speedup of backtracking algorithms. *arXiv preprint arXiv:1509.02374*.

Montanaro, A. (2020). Quantum speedup of branch-and-bound algorithms. *Physical Review Research*, 2(1):013056.

Mosseri, R. and Dandoloff, R. (2001). Geometry of entangled states, Bloch spheres and Hopf fibrations. *Journal of Physics A: Mathematical and General*, 34(47):10243.

Nagarajan, H., Lockwood, O., and Coffrin, C. (2021). QuantumCircuitOpt: An open-source framework for provably optimal quantum circuit design. In *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, pages 55–63. IEEE.

Nannicini, G. (2019). Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E*, 99(1):013304.

Nannicini, G. (2021). Fast Quantum Subroutines for the Simplex Method. In Singh, M. and Williamson, D. P., editors, *Integer Programming and Combinatorial Optimization - 22nd International Conference, IPCO 2021, Atlanta, GA, USA, May 19-21, 2021, Proceedings*, volume 12707 of *Lecture Notes in Computer Science*, pages 311–325. Springer.

Nannicini, G. (2024). Fast quantum subroutines for the simplex method. *Operations Research*, 72(2):763–780.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Nielsen, M. A. and Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

Nüßlein, J., Gabor, T., Linnhoff-Popien, C., and Feld, S. (2022). Algorithmic QUBO formulations for k-SAT and hamiltonian cycles. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2240–2246.

Oh, Y.-H., Mohammadbagherpoor, H., Dreher, P., Singh, A., Yu, X., and Rindos, A. J. (2019). Solving multi-coloring combinatorial optimization problems using hybrid quantum algorithms. *arXiv preprint arXiv:1911.00595*.

Omer, J., Poss, M., and Rougier, M. (2024). Combinatorial Robust Optimization with Decision-Dependent Information Discovery and Polyhedral Uncertainty. *Open Journal of Mathematical Optimization*. https://doi.org/10.5802/ojmo.33.

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and O'brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):1–7.

Pilon, G., Gugole, N., and Massarenti, N. (2021). Aircraft loading optimization–qubo models under multiple constraints. *arXiv preprint arXiv:2102.09621*.

Pinedo, M. L. (2012). *Scheduling*, volume 29. Springer.

Ploton, O. (2023). Contributions of inclusion-exclusion to exact or approximate solution of scheduling problems.

Ploton, O. and T'kindt, V. (2022). Exponential-time algorithms for parallel machine scheduling problems. *Journal of Combinatorial Optimization*, 44(5):3405–3418.

Ploton, O. and T'kindt, V. (2023). Moderate worst-case complexity bounds for the permutation flowshop scheduling problem using inclusion–exclusion. *Journal of Scheduling*, 26(2):137–145.

Poss, M. (2018). Robust combinatorial optimization with knapsack uncertainty. *Discrete Optimization*, 27:88–102.

Powell, M. J. (1994). *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2:79.

Pugliese, L. D. P., Guerriero, F., and Poss, M. (2019). The resource constrained shortest path problem with uncertain data: a robust formulation and optimal solution approach. *Computers & Operations Research*, 107:140–155.

Radzihovsky, M., Murphy, J., and Mason, S. (2019). A QAOA solution to the traveling salesman problem using pyQuil.

Ruan, Y., Marsh, S., Xue, X., Liu, Z., Wang, J., et al. (2020). The quantum approximate algorithm for solving traveling salesman problem. *Computers, Materials & Continua*, 63(3):1237–1247.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sack, S. H. and Serbyn, M. (2021). Quantum annealing initialization of the quantum approximate optimization algorithm. *quantum*, 5:491.

Shang, L., Lenté, C., Liedloff, M., and T'Kindt, V. (2018). Exact exponential algorithms for 3-machine flowshop scheduling problems. *Journal of Scheduling*, 21:227–233.

Shapiro, A. (2003). Monte Carlo sampling methods. *Handbooks in Operations Research and Management Science*, 10:353–425.

Shimizu, K. and Mori, R. (2022). Exponential-time quantum algorithms for graph coloring problems. *Algorithmica*, pages 1–19.

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

Soloviev, V. P., Larrañaga, P., and Bielza, C. (2022). Quantum parametric circuit optimization with estimation of distribution algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2247–2250.

Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341.

Tabi, Z., El-Safty, K. H., Kallus, Z., Hága, P., Kozsik, T., Glos, A., and Zimborás, Z. (2020). Quantum optimization for the graph coloring problem with space-efficient embedding. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 56–62. IEEE.

T'kindt, V., Della Croce, F., and Liedloff, M. (2022). Moderate exponential-time algorithms for scheduling problems. *4OR*, pages 1–34.

Vazirani, V. V. (2001). *Approximation algorithms*, volume 1. Springer.

Wang, Z., Hadfield, S., Jiang, Z., and Rieffel, E. G. (2018). Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A*, 97(2):022304.

Woeginger, G. J. (2003). Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 185–207. Springer.

Wurtz, J. and Love, P. (2021). Maxcut quantum approximate optimization algorithm performance guarantees for p¿ 1. *Physical Review A*, 103(4):042612.

Wurtz, J. and Love, P. J. (2022). Counterdiabaticity and the quantum approximate optimization algorithm. *Quantum*, 6:635.

Yang, Z.-C., Rahmani, A., Shabani, A., Neven, H., and Chamon, C. (2017). Optimizing variational quantum algorithms using Pontryagin's minimum principle. *Physical Review X*, 7(2):021027.

Yarkoni, S., Neukart, F., Tagle, E. M. G., Magiera, N., Mehta, B., Hire, K., Narkhede, S., and Hofmann, M. (2020). Quantum shuttle: traffic navigation with quantum computing. In *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*, pages 22–30.

Yuster, R. and Zwick, U. (2005). Fast sparse matrix multiplication. *ACM Transactions On Algorithms (TALG)*, 1(1):2–13.

Zhou, L., Wang, S.-T., Choi, S., Pichler, H., and Lukin, M. D. (2020). Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067.

Zhu, L., Tang, H. L., Barron, G. S., Calderon-Vargas, F., Mayhall, N. J., Barnes, E., and Economou, S. E. (2022). Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer. *Physical Review Research*, 4(3):033029.