



**UNIVERSITÉ
DE LORRAINE**

**BIBLIOTHÈQUES
UNIVERSITAIRES**

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4

Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

http://www.cfcopies.com/V2/leg/leg_droi.php

<http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm>

Quantum algorithms for energy management optimization problems

THÈSE

présentée et soutenue publiquement le 16 décembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine
(Informatique)

par

Margarita Veshchezerova

Composition du jury

<i>Président :</i>	Ioan Todinca
<i>Rapporteurs :</i>	Vedran Dunjko Caroline Prodhon
<i>Examineurs :</i>	Bernardetta Addis Philippe Lacomme Marc Porcheron
<i>Encadrants :</i>	Emmanuel Jeandel Simon Perdrix

Mis en page avec la classe thesul.

Remerciements

Three years on this thesis would be much less enjoyable without several people I would love to thank.

Firstly, I could not have undertaken this journey without my supervisors Emmanuel, Simon, and Marc. I'd like to thank them for the time they gave me, for their experience, and for all the advice that made this work better. I'm grateful for the large variety of topics they introduced to me, it was a pleasure to learn and manipulate many scientific concepts under their supervision.

I would like to extend my sincere thanks to Caroline and Verdran for reviewing my manuscript, and to Bernardetta, Philippe, and Ioan for being part of my defense committee.

Thanks should also go to my former office mate Vladimir and to other members of the (very big) MOCQUA team for the good time that I had in the lab. I gratefully acknowledge the help of my colleagues from EDF - Julien and Joseph.

Obviously, I can't forget to mention my parents and my closest friends - Vika, Sylvain, Charly, and Kiti. Their continuous support helped me to go through this thesis during its most satisfying and most challenging moments.

Finally, words cannot express my gratitude to Dinara and Pierre - for being at my side during these three years, for their encouragement in all moments of doubt, and for making me happy regardless of any difficulties unavoidable in a life of a Ph.D. student.

To my mother and grandmother,

Contents

Introduction	1
I Background	7
1 Quantum computing	9
1.1 Dirac notations	10
1.2 Qubits	10
1.2.1 Preparation and measurement	11
1.2.2 N-qubit systems	11
1.3 Quantum evolutions	13
1.3.1 Elementary gates	14
1.3.2 Compositions	16
1.4 Quantum program	17
2 Combinatorial Optimization	19
2.1 Decision, search and optimization problems	19
2.2 Complexity of optimization problems	20
2.2.1 NP-hardness	21
2.3 Approximation algorithms	23
2.3.1 Approximation ratio	24
2.3.2 APX	24
2.3.3 PTAS	25
2.3.4 FPTAS	25
2.3.5 Heuristics	26
2.4 Semidefinite programming	27

II	Quantum algorithms for Smart Charging problems	29
3	Quantum algorithms for combinatorial optimization	31
3.1	Motivation	31
3.1.1	Quantum acceleration of classical routines	31
3.1.2	Quantum heuristics	32
3.2	Ising Hamiltonian	36
3.2.1	Quadratic Unconstrained Binary Optimization (QUBO)	37
3.2.2	Beyond QUBO	39
3.3	Quantum Adiabatic Algorithm	40
3.3.1	Classical annealing	42
3.4	Quantum Annealing	43
3.4.1	Performance scaling for different annealing times	44
3.4.2	Experimental performance	45
3.5	Quantum Approximate Optimization Algorithm (QAOA)	46
3.5.1	QAOA components	47
3.5.2	How to compute the mean energy?	49
3.5.3	Performance guarantees for QAOA	51
3.5.4	QAOA circuit	51
3.5.5	Hardware implementations	53
3.5.6	Locality in QAOA	54
3.5.7	Parameter optimization	57
3.6	Recursive QAOA	61
4	Smart Scheduling	65
4.1	Problem statement	65
4.1.1	Complexity and approximations	66
4.1.2	Max-m-Cut reformulation	67
4.2	Max-m-Cut	68
4.2.1	The case $m = 2$	68
4.2.2	Complexity and approximations for $m > 2$	70
4.3	QAOA for Smart Scheduling	70
4.3.1	Encodings	70
4.4	Numerical results	73
4.4.1	Experimental setup	73
4.4.2	Establishing the protocol for QAOA	73
4.4.3	Performance evaluation	76

4.4.4	Possible improvements	80
5	Charging Task Selection	83
5.1	Problem statement	83
5.1.1	Group constraint	83
5.1.2	Conflict graph representation	84
5.2	Maximum Independent Set	85
5.2.1	Binary formulations of the M(W)IS problem	85
5.2.2	Different graph types	86
5.2.3	Complexity and approximations	87
5.2.4	Classical Heuristics	88
5.3	Quantum heuristics for Charging Task Selection	89
5.3.1	Encoding	89
5.3.2	Maximum Independent set on neutral atoms	90
5.3.3	Numerical results	92
6	Hybrid Quantum-Classical Decomposition Scheme	97
6.1	Integer Linear Programs	99
6.1.1	Linear relaxation	100
6.1.2	Branch & Bound	100
6.2	Classical Branch & Price	102
6.2.1	Dantzig-Wolfe decomposition	102
6.2.2	Decomposition for integer linear programs	103
6.2.3	ILP formulations with large number of variables	104
6.2.4	Column generation	105
6.2.5	Branch & Price for Integer Linear Programs	108
6.3	Quantum-assisted Branch & Price	110
6.3.1	Embedding (R)QAOA into the Branch & Price	110
6.4	Graph Coloring with Quantum-assisted Branch & Price	111
6.4.1	Column generation	112
6.4.2	A hybrid approach to the pricing subproblem	113
6.4.3	Numerical results	115
6.5	Discussion	115
III	Variational algorithms with ZX calculus	117
7	Introduction to ZX calculus	119

7.1	ZX diagrams	120
7.1.1	Generators	120
7.1.2	Compositions	122
7.2	Adjoint diagram	124
7.3	Universality	124
7.4	Rewrite rules	126
7.4.1	Meta-rule	126
7.4.2	Hadamard rules	127
7.4.3	Spider fusion	128
7.4.4	Bialgebra rules	128
7.4.5	Completeness	130
7.5	Linear diagrams	134
8	ZX-diagrams for variational quantum algorithms	137
8.1	The diagram for QAOA circuit	137
8.2	Loss function	139
8.2.1	The mean value of $\langle Z_i \rangle$	142
8.2.2	The mean value of $\langle Z_i Z_j \rangle$	144
8.3	Discussion	154
9	Addition and Differentiation of ZX-diagrams	155
9.1	Motivation	155
9.1.1	Addition in quantum computing	156
9.1.2	Differentiation in variational algorithms	158
9.2	Addition of ZX-diagrams	160
9.2.1	Controlled states	160
9.2.2	Controlizer	163
9.2.3	Relation to other results	169
9.3	Differentiation of ZX-diagrams	169
9.3.1	Diagrammatic differentiation with controlizers	170
9.3.2	Formula for derivatives in $ZX(\beta)$	175
9.3.3	Simplified formula for paired spiders	183
9.3.4	Relation to other results	187
9.4	Diagrammatic representation of Ising Hamiltonians	187
	Conclusion	189

A Linear programming	191
A.1 Duality	191
A.1.1 Complementary slackness	193
A.2 Simplex method	193
A.2.1 Graphical interpretation	195
Appendixs	191
Bibliography	197
A.3 Résumé étendu (en français)	221

List of Figures

1.1	Bloch sphere	12
1.2	Classical circuit	14
3.1	Hamiltonians for Adiabatic Algorithm, QAOA and Quantum Annealing	34
3.2	Quantum tunneling versus thermal hopping through a potential barrier	44
3.3	The interaction between classical and quantum parts of QAOA	50
3.4	Example of the optimization problem.	52
3.5	Example of the support of an observable	55
3.6	Support of the operator $Z_i Z_{i+1}$ after p layers of QAOA on the ring of disagree problem	56
3.8	Example of QAOA parameter landscape	58
4.1	Priorities p_j in the <i>Smart Scheduling</i> problem	66
4.2	Example of <i>smart-charging</i> instance for $m = 3$ an the corresponding instance of the Max-3-Cut problem	68
4.3	Concentration of optimal parameters for QAOA on the Smart Scheduling problem	74
4.4	Optimal QAOA parameters for different depths p for an instance of Smart Scheduling problem	75
4.5	Performance of different local search methods in the optimization of QAOA parameters	76
4.6	QAOA approximation ratio on the Smart Scheduling problem	77
4.7	Approximation ratios of QAOA and the random algorithm on the Smart Scheduling problem	77
4.8	Explaining the performance improvement of QAOA with size of the smart scheduling instances	80
4.9	Distribution of the maximum of Two Gaussian Random Variables	80
5.1	Instance of the Charging Task Selection problem	84
5.2	RQAOA versus a classical heuristic on Charge Task Selection problem.	94
6.1	Branch and bound tree	102
6.2	Quantum-classical column generation	108
6.3	Branch & Price	110
6.4	Hybrid algorithm for the pricing subproblem for graph coloring	114
7.1	Example of a ZX-diagram	120
7.2	Axioms for ZX	131
A.1	Simplex method	195

A.2	Schema d'interaction entre la partie classique et la partie quantique dans QAOA	223
-----	---	-----

Introduction

This thesis, co-funded and co-supervised by EDF R&D and LORIA, is dedicated to the application of quantum algorithms to combinatorial optimization problems issued from the field of energy management. We consider several optimization problems related to the charge of electric vehicles from the perspective of quantum heuristics such as *QAOA* and *Quantum Annealing*. From a theoretical perspective, this work presents new results derived for the graphical framework for quantum computing called ZX-calculus.

Smart charging

The field of energy management deals with a lot of complex optimization problems. Finding better solutions for these problems may significantly reduce costs, improve the quality of the service or solve such emergent operational problems as the integration of renewable energy sources.

Recent growth of the number of electric vehicles creates new challenges as well as additional opportunities for electricity management.

On one side, new *problems* emerge such as charging task attribution, scheduling, cost optimization (among many other). Indeed, even the most powerful charging terminals supplied by *Tesla*¹ (with electric power going up to 250 kW) require around 30 minutes to complete a full charge that assures approximately 400 kilometers of autonomy. We remark that 30 minutes per charge is an extremely optimistic estimation as only 6% of the charging points in France supply more than 50 kW of electric power. The operational challenges are high as the number of vehicles has dramatically increased in recent times. For instance, according to the French National Association of Electrical Mobility *L'Avere-France*, in August 2022 there were 69428 public charging points and more than 500000 electric vehicles in France [ave, 2022].

The growing number of electric vehicles puts an important pressure on the electrical production and distribution. For instance, in 2019 an average annual electricity consumption was about 4792 kWh per housing [Dupret *et al.*, 2021]. This statistic was evaluated on a selection of 100 housings one of which was equipped with an electric vehicle. In one year the vehicle in question consumed 2782 kWh, i.e. more than a half of the electrical consumption of an average housing.

On the other side, electric vehicles may *positively contribute* to the electricity management by improving existing solutions for the *storage* and the *peak load management*. Indeed, in a *vehicle to grid (V2G)* model electric vehicles act not only as consumers, but also as potential suppliers of the energy. For instance, the battery may be used to store the energy during the production peaks (particularly pertinent for renewable sources) and to return it to the grid afterwards. Such scenarios may improve the flexibility of the electrical system, reduce high-peaks, and thus generate significant energy savings, while offering customers with various services involving "*earning money*" and "*saving costs*" opportunities.

¹https://www.tesla.com/en_eu/supercharger

The field of optimization problems related to the charging of electric vehicles is known by the name of *smart charging*.

Computational hardness of operational problems

A lot of problems of the electricity management in general and smart charging in particular are modeled as NP-hard combinatorial optimization problems. The NP-hardness implies that (at least in the worst case) the search for the optimal solution takes presumably exponential runtime. Therefore, in practice such NP-hard problems are solved only *approximately*. An approximate solution may be computed by two kinds of algorithms: the ones with a *theoretically-proven approximation ratio* and the algorithms (called *heuristics*) that have demonstrated good empirical results.

In the context of energy management even a minor improvement of the solution's quality may significantly reduce the costs and increase profits, so the active search for better algorithms never ceases.

Quantum approaches for combinatorial optimization

For computationally difficult problems the rapidly growing field of quantum computing inspires many hopes. Indeed, machines that exploit the principals of quantum mechanic such as *entanglement* and *superposition* are provably more powerful than classical Turing machines. Therefore, we expect quantum algorithms to provide significant speedups for some important problems. For the moment, two most impressive results are *Grover's algorithm* for a search in an unstructured database [Grover, 1996] and the algorithm for integer factorization discovered by Shor [Shor, 1995]. While Grover's algorithm guarantees a *quadratic speedup* over the *best possible* classical algorithm, the speedup of Shor's algorithm is *exponential* over the *best known* classical counterpart.

The major question is, however, if quantum computing will ever become a reality. Recent progress of multiple hardware teams is very encouraging. For the moment, the biggest *universal quantum computer* is the 127-qubit machine *IBM Eagle* [Dial, 2022]. Google's Sycamore processor has 54 superconducting qubits [Arute *et al.*, 2019]. The biggest analogous quantum annealer with 5000 qubits is the *Advantage system* constructed by DWave [Systems, 2022]. In France the quantum hardware is manufactured by such companies as *Pasqal*² that develops a platform on *neutral atoms*, *Quandela*³ that is based on *photonic qubits* or *Alice & Bob*⁴ working on *superconducting qubits*. A comprehensive review of the quantum hardware and software ecosystems can be found in [Ezratty, 2021].

On the down side, the experimental progress testifies how difficult it is to robustly scale quantum machines. Therefore, we are not expecting to see a big fault-tolerant computer in the close future. Instead, we can reasonably hope to access in next years Noisy Intermediate Scale devices (called NISQ devices) that will operate around 10^2 - 10^3 qubits. Due to their limited size and high noise level, we don't expect NISQ machines to execute famous Grover's and Shor's algorithms on problems of practically interesting sizes [Babbush *et al.*, 2021]

There exist, however, quantum algorithms that do not require deep circuits and heavy error-correction and, as a consequence, are well adapted for NISQ [Preskill, 2018]. For combinatorial optimization we are speaking about such heuristics as VQE, QAOA and Quantum Annealing. In

²<https://pasqal.io/>

³<https://www.quandela.com/>

⁴<https://alice-bob.com/>

addition, we believe that the right way to exploit the power of NISQ is to use *hybrid quantum-classical procedures* that integrate quantum subroutines in some classical algorithm. Following this direction, in this thesis we introduce *Quantum-assisted Branch & Price* - an algorithm for huge integer programs that integrate quantum heuristics in the classical Branch & Price framework.

From the application side, the goal of this thesis is to explore the potential of NISQ-adapted algorithms as well as hybrid procedures on real-world optimization problems. We don't expect to provably solve NP-hard problems to optimality, but rather qualify quantum algorithms as heuristics. We remark that this ambition is limited by the hardware availability as well as its power. Indeed, both current quantum machines and most performant classical emulators such as the *QLM* developed by *Atos* [QLM, 2022] can address only relatively simple models of very modest sizes (around 40 variables).

Qualification of quantum algorithms for smart charging problems

Most conventional quantum heuristics for combinatorial optimization accept the input formulated as *quadratic unconstrained binary optimization (QUBO)* problems. This means that a special treatment is required to capture in a QUBO model such widely-spread things as constraints and non-binary variables. Proper *modelization* is indeed a fundamental challenge as many natural formulations of real-world optimization problems exceed the QUBO framework. It turns out that sometimes in order to get a reasonable QUBO formulation one has to significantly reduce of the complexity of the initial model.

We remark that there is an additional difficulty that is usually ignored in the qualification of quantum algorithms. This difficulty is the choice of a *relevant* classical competitor. Indeed, while an NP-hard optimization problem in general can't be efficiently solved to optimality, there may be very efficient methods that return suboptimal but still satisfying solutions. On the other hand, for many NP-hard problems there exist inapproximability results that bound the theoretically achievable approximation ratio.

In this thesis we consider two real-world problems issued from the domain of electric vehicles. The first problem relates to the *scheduling*: the goal is to assign a set of charges to different charging stations while minimizing the weighted completion time. In the second problem one has to select a subset of charge demands to satisfy while respecting conflict between demands.

In sections dedicated to numerical results we provide some insights about the experimental protocol for numerical simulations. Our algorithms are tested on realistic data generated from the *Belib dataset* [Bel, 2017] and on a dataset produced by a simulator of charging demands developed by EDF. We compare the performance of our quantum routines to carefully selected classical counterparts.

The work dedicated to the application of quantum heuristics to smart charging problems was previously reported in our paper [Dalyac *et al.*, 2021] and is a part of the current thesis (chapters 4 and 5).

ZX-calculus for variational algorithms

In parallel with experimental evaluation of variational algorithms we considered the ways to enhance the theoretical research in the field. For this purpose we decided to use the ZX calculus - a powerful framework that allows to graphically reason about quantum computing.

In ZX-calculus, originally introduced in [Coecke and Duncan, 2011], quantum computations are represented by ZX-diagrams. Each ZX-diagram may be interpreted as a linear map. Conversely, for every linear map $M : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^m}$ there exists a corresponding ZX-diagram. The diagrammatic representation, however, is not unique, so the ZX-calculus comes with a set of rewrite rules - local graphical transformations that preserves the matrix interpretation of the diagram.

An introduction to the ZX-calculus is provided in the chapter 7.

The ZX-calculus was shown to be extremely helpful in many domains of quantum computing. It was, however, rarely applied in the analysis of variational algorithms such as QAOA. We believe that the reason why variational algorithms are still unexplored with the means of the ZX-calculus is the absence of a convenient way to add and differentiate diagrams. **For this reason we introduced in our paper [Jeandel *et al.*, 2022] several original techniques that allow diagrammatic addition and the differentiation.** We present these techniques in chapter 9.

Organization of the manuscript

This document contains three parts.

The first part (I) provides the necessary background in quantum computing and combinatorial optimization. For instance, in chapter (1) that introduces quantum computing we present the concepts of *qubit* and *unitary transformation* as well as the *circuit notation* for quantum programs.

In the following chapter (2) we introduce the complexity notions for combinatorial optimization problems that are relevant for comparisons of quantum and classical methods. In addition, in section 2.4 we provide a brief introduction to the *semidefinite programming* often used in the design of approximation algorithms.

The second part (II) is dedicated to the application of quantum heuristics to smart charging problems.

It begins with the chapter (3) that introduces quantum algorithms for combinatorial optimization. We present such algorithms as Adiabatic Algorithm (section 3.3), Quantum Annealing (section 3.4), QAQA (section 3.5) and RQAOA (section 3.6). We provide an extensive review of the bibliography on considered algorithms. Based on this bibliography, we compare the expected performance and applicability of the presented methods from the theoretical and practical points of view.

Concerning our contribution, in section 3.5.1 we present our analytical formula (3.37) for the energy expectation in QAOA₁ on instances of the weighted MaxCut problem. We also provide our original proof for the bound on approximation ratio achievable by QAOA on the *Ising chain model* (claim 3.5.2).

The presentation of quantum algorithms for optimization is pursued with examples of their application to the *smart scheduling* problem (chapter 4) and the *charge task selection* problem (chapter 5). For both considered applications we present the original usecase formulations, the complexity analysis, the modelization process as well as numerical results that are compared to the performance of classical methods. For the modelization part we introduce in the section 4.3.1 an original *binary encoding technique* allowing the integration of bounded integer variables in the QUBO framework. Finally, we compare our quantum algorithms to carefully chosen classical counterparts in order to place our results in the context of the qualification of quantum approaches on real-world applications.

In the following chapter (6) we present an original hybrid quantum-classical algorithm for

huge integer programs that was developed during this thesis. The method is based on the column generation approach presented in section 6.2.4. We baptize our hybrid method under the name of *Quantum-assisted Branch & Price*. Quantum-assisted Branch & Price can be applied to integer programs with a large number of variables as ones obtained from Dantzig-Wolfe decomposition (see section 6.2.1). In section (6.4), we illustrate our approach on an example of *graph coloring* problem. We numerically evaluate the performance of our method on a version of the *charge task selection* problem.

The last part (III) presents the application of the ZX-calculus (introduced in chapter 7) to variational algorithms. In chapter (8), we show how QAOA routines can be represented with ZX-diagrams. We illustrate the process of graphical rewriting by showing how to derive an analytical expression for the loss function optimized by the classical loop of QAOA. This proof is presented in the current thesis for the first time.

In the final chapter (9) we describe our original techniques for addition (section 9.2) and differentiation (section 9.3) of ZX-diagrams. In section (9.4) we demonstrate how these techniques can be used to derive a ZX-diagram for an Ising Hamiltonians.

In order to make the work as comprehensible as possible, we include the appendix A that introduces the basic concepts of *linear programming*.

Part I

Background

Chapter 1

Quantum computing

The idea to use quantum effects to enhance the computations of properties of physical systems (and computation in general) was independently suggested by Richard Feynman [Feynman, 1986], Yuri Manin [Manin, 1980] and Paul Benioff [Benioff, 1982]. This idea is particularly relevant in the context of the simulation of quantum systems. Indeed, according to the laws of quantum mechanics such system behaves differently from what could be expected in classical physics. In particular, a quantum system can be "*in many states at once*" - a phenomenon known as *quantum superposition*. An explicit enumeration of a superposition state requires an exponential amount of complex coefficients. Therefore, the simulation rapidly becomes intractable when the size of the quantum system grows. In other words, the *nature is quantum* and classical computing models are not powerful enough to efficiently compute properties of quantum systems.

On the practical side, the miniaturization process that made the *classical computer* so powerful, can't continue at the previously-observed exponential rate reclaimed by the famous empirical *Moore law* [Wikipedia, 2022b]. The limitation is due, inter alia, to quantum effects that have a significant impact on the dynamics of small systems. In opposition, quantum computers are physical systems that, rather than considering quantum effects as a resource of undesirable noise, use them as an opportunity to compute differently. Somewhat tautologically, quantum computing can be defined as the domain of computer science that explores the computational power of such quantum computers. The quantum computational model was formalized in a definition of universal quantum Turing machines in [Deutsch, 1985].

Early algorithms for quantum computers such as Deutsch-Jozsa [Deutsch and Jozsa, 1992] and Simon's algorithms [Simon, 1997] were mostly of theoretical interest. It was the procedure for an efficient integer factorization suggested by Shor [Shor, 1995] that attracted a lot of attention to quantum computing.

This chapter provides a brief introduction to the basic concepts of quantum computing. We consider, in particular, the *digital model* of quantum computing that is realized on universal machines. Analogue quantum machines supporting adiabatic quantum computing are presented in chapter 3.

A seminal reference for an introductory course in the domain of quantum computing is [Nielsen and Chuang, 2011]. Alternatively, the lecture notes [de Wolf, 2019] provides a comprehensive and accurate presentation of the field of quantum computing and related domains.

1.1 Dirac notations

In a nutshell, a quantum algorithm is a transformation of a vector from the space \mathbb{C}^{2^n} by unitary matrices from $\mathbb{C}^{2^n \times 2^n}$ followed by a probabilistic sampling (we denote by $\mathbb{C}^{a \times b}$ a complex matrix with a columns and b rows). In order to reason about vectors and matrices, most modern works on quantum mechanics and quantum computing use the so-called *Dirac notation*.

In this notation we call by "ket" a column vector:

$$|\psi\rangle = \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_n \end{pmatrix} \quad (1.1)$$

and by "bra" a row vector:

$$\langle\psi| = (\psi_0^*, \dots, \psi_n^*) \quad (1.2)$$

with complex numbers as elements. We can get a *bra-vector* from the *ket-vector* by taking its conjugate transpose.

We denote by dagger v^\dagger (alternatively called *adjoint*) the transposition and complex conjugation, i.e. for $v = \begin{pmatrix} v_0 \\ \vdots \\ v_n \end{pmatrix}$ we have $v^\dagger = (v_1^*, \dots, v_n^*)$. Certainly, for the ket-state $|\psi\rangle$ the adjoint

is the bra-state $\langle\psi|$. For a linear map $M : \mathbb{C}^{n \times m}$ its adjoint $M^\dagger : \mathbb{C}^{m \times n}$ is similarly defined as a combination of the transposition and the complex conjugation.

Using the "bra" and "ket" notations we can express the norm of a column vector $|\psi\rangle$ in the Hilbert space \mathbb{C}^n as a *bracket* $\langle\psi|\psi\rangle$. We recall that in the Hilbert space the *inner product* of vectors $\langle\phi|$ and $|\psi\rangle$ is precisely $\sum_{i=1}^n \phi_i^* \psi_i$. We denote the inner product with the bracket $\langle\phi|\psi\rangle$.

Following the widely-adopted conventions, we denote by $|0\rangle$ and $|1\rangle$ the standard basis in a two-dimensional complex space \mathbb{C}^2 :

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (1.3)$$

The basis in the space \mathbb{C}^{2^n} of dimension 2^n is usually enumerated by bitstrings $x \in \{0, 1\}^n$.

1.2 Qubits

In classical computing, the information is carried by *bits*. A bit is a classical system that is in one of two mutually exclusive states denoted by 0 and 1. Crucially, at each moment a classical system is *strictly* in one state. We remark that in probabilistic algorithms some bits $b \in \{0, 1\}$ can be sampled from a probability distribution $p \in \mathbb{R}^n$ given by a vector of *real values* with the 1-norm $\sum_{i=1}^n p_i = 1$ while in quantum computing the information is carried by a vector $|\psi\rangle$ of *complex numbers* with the *Hermitian norm* $\langle\psi|\psi\rangle = 1$. In addition, once a classical bit is assigned a random value it remains in the same (potentially unknown) state unless it is explicitly reassigned. In quantum computing, a simple observation of the system actually modifies the state of the system.

A classical algorithm can interact with the memory in two ways. Firstly, we can assign a value $v \in \{0, 1\}$ to the bit $b := v$. An inverse procedure outputting the value of the bit $o := b$ is called *reading*.

In quantum computing, the information is contained in a state of some quantum system. We consider systems that have two distinct elementary states $|0\rangle$ and $|1\rangle$. The simplest example of a such two-dimensional quantum system is the spin of a particle that can be turned up or down: $|0\rangle = |\uparrow\rangle$ and $|1\rangle = |\downarrow\rangle$. A two-level quantum system is called **qubit**.

In mathematical terms, state of a qubit $|\psi_0\rangle \in \mathbb{C}^2$ is a vector of two complex numbers a and b often called *amplitudes*:

$$|\psi_0\rangle = a|0\rangle + b|1\rangle \quad (1.4)$$

$$|a|^2 + |b|^2 = 1 \quad (1.5)$$

We observe that the only restriction on amplitudes is the condition (1.5) stating that the norm of the vector should be equal to one. Therefore, a state of a qubit can be any vector on the unit sphere \mathbb{S}^2 in the two-dimensional complex space. If both amplitudes a and b are non-zero we say that the qubit is in a *superposition* of basis states.

1.2.1 Preparation and measurement

Similarly to *reading* and *writing* for classical bits, there exist two ways to interact with a qubit called *preparation* and *measurement*.

Preparation in a trivial state is analogous to the writing for bits:

$$|\psi\rangle = |0\rangle \quad (1.6)$$

The reading part, called *measurement*, is slightly less intuitive. The complexity is due to the fact that *we are classical*, so we can access only classical information about the quantum system. More precisely, the measurement of a qubit returns a classical output $o \in \{0, 1\}$ which is a binary number. In such settings, the quantum state $|\psi\rangle = a|0\rangle + b|1\rangle$ defines the *probability* to measure a specific value:

$$P(o = 0) = |a|^2 \quad (1.7)$$

$$P(o = 1) = |b|^2 \quad (1.8)$$

Crucially, two subsequent measurements without modifications in-between should return *exactly the same* output $|o\rangle$. Therefore, the measurement "*collapses*" the superposition state $|\psi\rangle = a|0\rangle + b|1\rangle$ to the basis state $|o\rangle$ corresponding to the measurement result. We highlight that after the measurement the initial superposition is irreversibly lost. Moreover, the measurement returns a binary value and the coefficients a and b are never directly accessible to the classical spectator.

We remark that the probability distribution is defined by the squared norms of amplitudes. This fact explains why the state should have a unit norm, i.e. why $|a|^2 + |b|^2 = 1$. We also notice that the multiplication by a global complex phase $|\psi\rangle \rightarrow e^{i\alpha}|\psi\rangle$ has no influence on the distribution, so we can safely ignore it. Therefore, the qubit state has precisely two degrees of freedom: one for each angle β and γ in the expression $|\psi\rangle = \cos\beta|0\rangle + e^{i\gamma}\sin\beta|1\rangle$. As the state has two degrees of freedom it can be visualized on a *Bloch sphere* shown in figure 1.1

1.2.2 N-qubit systems

Tensor product

For each pair of vectors $|v\rangle$ and $|u\rangle$ we introduce the shortcut notation:

$$|v\rangle|u\rangle = |v\rangle \otimes |u\rangle \quad (1.9)$$

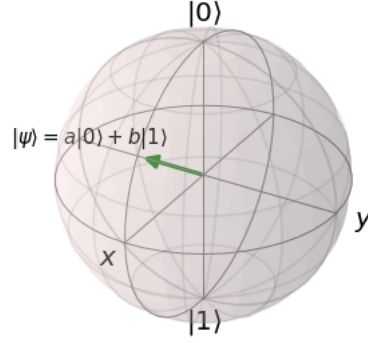


Figure 1.1: The qubit state $|\psi\rangle = \cos \beta |0\rangle + e^{i\gamma} \sin \beta |1\rangle$ on the Bloch sphere. The figure is plotted with the python package QuTIP [<https://qutip.org>]

where \otimes is the *tensor product* of vectors also sometimes called *Kronecker product*. We recall that for vectors $|v\rangle = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{C}^n$ and $|u\rangle = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \in \mathbb{C}^m$ the tensor product $|v\rangle \otimes |u\rangle$ is a vector in \mathbb{C}^{nm} of the form:

$$|v\rangle \otimes |u\rangle = \begin{pmatrix} v_1 \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \\ \vdots \\ v_n \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \end{pmatrix} \quad (1.10)$$

For the matrices $A \in \mathbb{C}^{n \times m}$ and $B \in \mathbb{C}^{k \times l}$ the tensor product is similarly defined as

$$A \otimes B = \begin{pmatrix} a_1^1 B & \dots & a_1^n B \\ & \ddots & \\ a_m^1 B & \dots & a_m^n B \end{pmatrix} \quad (1.11)$$

Entanglement

Trivially, a state of n classical bits $\mathbf{b} \in \{0, 1\}^n$ is in a one-to-one correspondence with n states of n bits $[b_0, \dots, b_{n-1}]$. A peculiar fact about quantum computing is that *in general* a state of n qubits can't be decomposed to n independent states of qubits $|\psi_0\rangle \dots |\psi_{n-1}\rangle$.

Indeed, a state of n qubit system $|\psi\rangle$ lays on the unit sphere in the Hilbert space \mathbb{C}^{2^n} and not in the direct-sum of n separated 2-dimensional Hilbert spaces $\mathbb{C}^2 \oplus \dots \oplus \mathbb{C}^2$. This implies in particular that some states of an n -qubit quantum register can't be decomposed on a tensor product of n individual one-qubit states. Such phenomenon is called *entanglement* and it is believed to be a fundamental source of "quantumness" in quantum computing.

As in the one-qubit case, the state $|\psi\rangle$ defines a probability distribution. Formally, the state in \mathbb{C}^{2^n} can be written as a linear combination of basis states $|x\rangle$, $x \in \{0, 1\}^n$:

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \quad (1.12)$$

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1 \quad (1.13)$$

where $|\alpha_x|^2$ corresponds to the probability to measure the output $|x\rangle$. As the basis states $|x\rangle$ are orthogonal, the coefficients α_x correspond to the inner product $\alpha_x = \langle x | \psi \rangle$.

The equality (1.13) can be alternatively written as:

$$\langle \psi | \psi \rangle = 1 \quad (1.14)$$

The simplest example of an entangled state is the *Bell state*:

$$|\phi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (1.15)$$

We can verify that there don't exist $|\phi_0\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$ and $|\phi_1\rangle = \begin{pmatrix} c \\ d \end{pmatrix}$ such that $|\phi\rangle = |\phi_0\rangle|\phi_1\rangle$. Indeed, $|\phi_0\rangle|\phi_1\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$ and the requirement $ad = bc \equiv 0$ contradicts to the fact that $ac = bd \equiv \frac{1}{\sqrt{2}}$.

1.3 Quantum evolutions

In quantum algorithm, qubit states are transformed by linear maps:

$$|\psi\rangle \xrightarrow{U} |\psi'\rangle = U|\psi\rangle \quad (1.16)$$

In order to obtain valid resulting states the transformation should map the unit sphere \mathbb{S}^{2^n} to itself, i.e. be norm-preserving. This requirement is satisfied by so-called *unitary matrices*.

Unitary matrices can be equivalently defined as square matrices for which we have:

$$U^\dagger U = I \quad (1.17)$$

In other words, the inverse of U is precisely the adjoint U^\dagger of U . Moreover, the inverse of a unitary matrix is also unitary i.e. *quantum computing allows only invertible transformations*. The only exception to this rule is the *measurement process* that irreversibly collapses the superposition $|\psi\rangle = \sum a_x |x\rangle$ to the measurement output $|\hat{x}\rangle$.

A transformation U is fully defined by the matrix of the corresponding linear map:

$$M_U \in \mathbb{C}^{2^n \times 2^n} \quad (1.18)$$

An explicit matrix representation for unitary transformations is extremely inefficient and resource-consuming. It is also impractical for programming, as we can't use the matrix to specify a set of instructions that have to be executed on the quantum computer. In classical computing of boolean functions, the analog of the matrix representation are the truth tables. We remark that truth tables and matrices are never used for large computations as there exist much more convenient representations that use *circuits*.

Circuit representation

A classical circuit is a graphical scheme that has *i*) input wires *i*) output wires and *iii*) elementary blocks conventionally called *gates*. Crucially, a small set made out of elementary AND-gate, OR-gate, and NOT-gate is powerful enough to express any boolean function as a circuit made out of them.

For example, we can represent the function $f(\mathbf{b}) = b_2 \vee (b_0 \wedge b_1)$ with the circuit shown in figure 1.2.

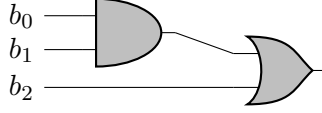


Figure 1.2: Classical circuit for $f(\mathbf{b}) = b_2 \vee (b_0 \wedge b_1)$

The idea to use circuits was borrowed by quantum computer science. A *wire* in a quantum circuit corresponds to a *qubit*. A circuit for the unitary evolution U is denoted as:



In a quantum circuit model, unitary transformations (just as boolean functions in classical circuits) are written as compositions of some elementary gates. The only fundamental difference with the classical case is that for purely quantum gates the number of inputs is *always* equal to the number of outputs. This condition directly follows from the reversibility of unitary quantum transformations.

A typical quantum circuit looks like the following:



In a circuit representation, we often omit the measurements as we usually assume that they are performed at the end of the circuit. If it is not the case (for instance in protocols that use ancilla qubits) we explicitly integrate the non-trivial measurement in the circuit.

1.3.1 Elementary gates

We briefly present some most commonly used gates.

Hadamard gate

So-called Hadamard gate is probably the most famous one-qubit transformation.

$$-\boxed{H}- = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (1.21)$$

The Hadamard gate maps the computational basis states $|0\rangle$ and $|1\rangle$ to a pair of orthogonal states $|+\rangle$ and $|-\rangle$:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \quad (1.22)$$

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \quad (1.23)$$

The pair $\{|+\rangle, |-\rangle\}$ with $|+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $|-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ is usually called the *Hadamard basis*. Alternatively, in some cases the bases $\{|0\rangle, |1\rangle\}$ and $\{|+\rangle, |-\rangle\}$ are respectively called *Z-basis* and *X-basis*.

Pauli gates

For the reader with physical background *Pauli gates* should remind of *Pauli matrices* used to represent spin interactions in quantum mechanics.

The *X*-gate is equivalent to the classical NOT gate. For the states $|0\rangle$ and $|1\rangle$ it performs the *bit flip*: $|b\rangle \rightarrow |\bar{b}\rangle$. The matrix for *X* is:

$$\text{---}\boxed{X}\text{---} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (1.24)$$

The *Z*-gate performs the so-called *phase flip*: $|b\rangle \rightarrow (-1)^b |b\rangle$. The matrix for the *Z*-gate is

$$\text{---}\boxed{Z}\text{---} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (1.25)$$

We remark that vectors from computational basis $\{|0\rangle, |1\rangle\}$ are eigenvectors for the *Z*-gate while the vectors $|+\rangle$ and $|-\rangle$ are eigenvectors for the *X*-gate. This is by the way the reason why these bases are sometimes called *Z-basis* and *X-basis*.

Matrices *X* and *Z* have real-values eigenvectors. This is not the case for the last Pauli gate called *Y*-gate:

$$\text{---}\boxed{Y}\text{---} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (1.26)$$

Pauli gates *X*, *Y* and *Z* are Hermitian and involutory ($P^2 = I$ for $P = X, Y, Z$). Moreover, together with the identity *I* they form a basis in the vector space of 2×2 Hermitian matrices.

Rotation gates

A rotation gate is actually not a single matrix but rather a family of gates parameterized by a real value γ called *rotation angle*.

We denote by $R_X(\gamma)$ the rotation around *X*-axis:

$$\text{---}\boxed{R_X(\gamma)}\text{---} = e^{i\gamma X/2} = \begin{pmatrix} \cos(\frac{\gamma}{2}) & i \sin(\frac{\gamma}{2}) \\ i \sin(\frac{\gamma}{2}) & \cos(\frac{\gamma}{2}) \end{pmatrix} \quad (1.27)$$

The rotation around *Z*-axis is:

$$\text{---}\boxed{R_Z(\gamma)}\text{---} = e^{i\gamma Z/2} = e^{i\frac{\gamma}{2}} \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\gamma} \end{pmatrix} \quad (1.28)$$

We recall that the matrix exponential $e^{i\gamma A}$ is the power series:

$$e^{i\gamma A} = \sum_{k=0}^{\infty} \frac{(i\gamma)^k A^k}{k!} \quad (1.29)$$

For involutory matrices $A^k = \begin{cases} I, & \text{if } k = 0 \pmod{2} \\ A, & \text{otherwise} \end{cases}$, so the power series (1.29) can be written as $e^{i\gamma A} = \cos \gamma I + i \sin \gamma A$.

We remark that for the rotation angle π the gates $R_X(\pi)$ and $R_Z(\pi)$ are precisely the Pauli gates X and Z up to a global phase.

Another important rotation gate is the so-called *T-gate* $T = R_Z(\pi/4)$. Together with Pauli gates, Hadamard gate, and a single two-qubit transformation the T-gate form an approximately universal set of gates for pure-qubit quantum computing.

CNOT gate

Interesting quantum effects such as *entanglement* occur when qubits interact with each other. A basic multiqubit gate is the two-qubit controlled NOT gate called CNOT. Intuitively, the CNOT gate does nothing with the state if the first qubit is in the $|0\rangle$ -state while it flips the value of the second qubit if the first one is in $|1\rangle$ state:

$$|0\rangle|b\rangle \xrightarrow{\text{CNOT}} |0\rangle|b\rangle \quad (1.30)$$

$$|1\rangle|b\rangle \xrightarrow{\text{CNOT}} |1\rangle|\bar{b}\rangle \quad (1.31)$$

The matrix representation for CNOT is:

$$\begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \oplus \\ \text{---} \end{array} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (1.32)$$

In the circuit notation it is represented by a wire that connects two different dots - one corresponding to the *control* and the other to the *target* qubit.

1.3.2 Compositions

Two elementary gates can be applied in parallel. The underlying transformation of parallel composition corresponds to the *tensor product* of individual unitaries:

$$\begin{array}{c} \text{---} \boxed{\mathcal{A}} \text{---} \\ \text{---} \boxed{\mathcal{B}} \text{---} \end{array} = A \otimes B \quad (1.33)$$

A *sequential composition* of gates implements the unitary equal to the *multiplication* of individual blocks:

$$\begin{array}{c} q_1 \text{ ---} \\ q_2 \text{ ---} \\ \vdots \\ q_n \text{ ---} \end{array} \begin{array}{|c|} \hline \mathcal{A} \\ \hline \end{array} \begin{array}{|c|} \hline \mathcal{B} \\ \hline \end{array} = B \circ A \quad (1.34)$$

We remark that in circuits the time evolves from left to right.

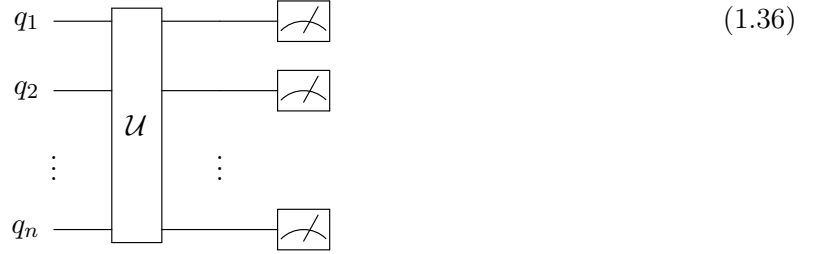
The gate set H , $CNOT$, and $R_Z(\gamma)$ is universal for quantum computation. Strictly speaking, one can write an arbitrary unitary evolution with a circuit containing only these gates. We remark that for each matrix the decomposition is usually not unique, for example two circuits:

$$\text{---} \boxed{H} \text{---} \boxed{Z} \text{---} \boxed{H} \text{---} \quad \text{and} \quad \text{---} \boxed{X} \text{---} \quad (1.35)$$

represent the same matrix $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

1.4 Quantum program

In nutshell, a quantum program prepares a system in a trivial state $|\psi_0\rangle = |0 \dots 0\rangle$ to which it applies a unitary transformation U given by a circuit. In the end the state $U|\psi_0\rangle$ is measured and a classical bitstring is recovered:



Just before the measurement the quantum state is:

$$|\psi\rangle = U|0 \dots 0\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \quad (1.37)$$

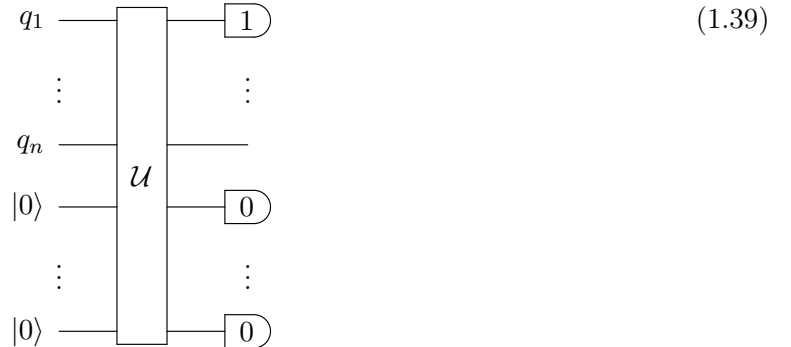
Measurement returns a sample from probability distribution:

$$\mathbb{P}(x) = |\alpha_x|^2, \quad x \in \{0,1\}^n \quad (1.38)$$

More complicated distributions can be prepared using auxiliary *ancilla qubits* and *postselection*.

Indeed, it is possible to consider additional ancilla inputs initialized in states $|0\rangle$. Ancillas allow us to perform the linear quantum evolution in a bigger space. One can also use a *postselection* on the measurement output, for example, keep only outputs where the first qubit takes value the 1. We remark that the post-selection, while possible in computing, is not a physical process.

Ancillas and postselection lead to non-unitary transformations. Nevertheless, there exist blocks permitting to integrate them inside the circuit notations:



From a very general point of view programming for a quantum computer consists in combining *small bricks* to prepare a *probability distribution* that is relevant for the problem in question. Certainly, it is a stiff exercise to design algorithms in such a framework. The challenge gets even more difficult when in addition we aim to find quantum algorithms that have a *provable advantage* over classical counterparts.

In part due to the difficulty of designing quantum algorithms the field of quantum computing transferred a big switch of paradigm moving to *variational algorithms*. In a nutshell, variational algorithms are not fixed programs solving a problem but rather trainable models that are optimized to find good solutions. We come back to the variational algorithm in chapter 3.

Chapter 2

Combinatorial Optimization

Combinatorial optimization is the field of mathematics that considers problems over discrete variables. Models with discrete variables are ubiquitous in many real-life applications such as scheduling, resource allocation, logistics, and others. From the formal side, combinatorial optimization incorporates such important models as constraint satisfaction problems, problems on graphs and other discrete structures as well as general optimization problems over integer variables.

Many problems of combinatorial optimization are computationally difficult. In the first part of the current chapter we formally define what is the computational difficulty of an optimization problem. In addition, we briefly introduce another important concept of *approximation complexity*. Most of our definitions are formulated as in the book [Ausiello *et al.*, 1999].

The second part of the chapter is dedicated to semidefinite programs (Section 2.4). Strictly speaking, semidefinite programs are optimization problems over *continuous* variables. However, they are extremely important in the design of approximation algorithms for combinatorial problems.

2.1 Decision, search and optimization problems

Optimization problems are closely related to *decision problems* and *search problems*. The relation to decision problems is particularly meaningful for the complexity analysis of the optimization problems. Indeed, *NP-completeness* which is the fundamental notion in the complexity theory, is defined in the framework of decision problems.

An optimization problem as well as decision and search problems are provided with a set of instances I and solutions S . A predicate $\pi : I \times S \rightarrow \{0, 1\}$ indicates if $s \in S$ is a solution for an instance $i \in I$.

- *Decision problems.* The solution set contains only two elements $S = \{0, 1\}$. For every instance $i \in I$ we have $\pi(i, 0)$ or $\pi(i, 1)$. In other words, the predicate π splits the set $I = I_p \sqcup I_n$ on positive and negative instances. The problem is to decide for a given instance i if $i \in I_p$ or $i \in I_n$.
- *Search problems.* To each instance i corresponds a set of feasible solutions $F(i) = \{y \in S \mid \pi(i, y) = 1\}$. The goal is to find for a given input instance a feasible solution y^* such that $\pi(i, y^*) = 1$.

- *Optimization problems.* For each instance feasible solutions are assigned to a score. The score is typically computed with so-called *measure function* $m : i \times F(i) \rightarrow m_i \in \mathbb{Z}$. We are looking for a feasible solution with the highest (or lowest) score value.

Formally, an optimization problem is defined by a tuple (I, F, m) and a *goal* which can be *maximization* or *minimization*. The set I contains instances of the problem. The map F associates to each valid instance i a set of feasible solutions $F(i)$. The function $m : I \times F(I) \rightarrow \mathbb{Z}$ is called *measure function*. The measure function associates to each pair (i, y) where $i \in I$ is an instance of the problem and $y \in F(i)$ is a feasible solution for the instance i a positive integer value. If the goal is to maximize the score, the measure function can be referred to as *profit function*. For minimization problems terms *cost function* and *loss function* are employed in the literature.

An optimal solution y^* to a maximization problem for a given instance $i \in I$ is a feasible solution $y \in F(i)$ of highest measure:

$$y^* \in F(i) \text{ s.t. } \forall y \in F(i) : m(i, y) \leq m(i, y^*) \quad (2.1)$$

The value $m^*(i) = m(i, y^*)$ is called the optimum value of the problem. For each instance the optimum value $m^*(i)$ is unique. Certainly, the set of optimal solutions $F^*(i) = \{y \in F(i) \mid m(i, y) = m^*(i)\}$ may contain several elements.

We illustrate the definition of an example of *graph coloring* problem. In *graph coloring* the set of instances I is the set of all graphs $G = (V, E)$. For each graph feasible solutions are proper colorings, i.e. functions $c : V \rightarrow \mathbb{N}$ such that connected vertices $(u, v) \in E$ are assigned to different colors: $c(u) \neq c(v)$. The *measure* or *cost* of coloring is the number of different colors, i.e. $m(i, c) = \#\{c \in \mathbb{N} \mid \exists u \in V : c(u) = c\}$.

Each optimization problem \mathcal{P} straightforwardly generates a decision problem \mathcal{P}_D . For instance, in the graph coloring example we may ask if there exists a coloring c that use less than k colors for a fixed number $k \in \mathbb{N}$. We remark that in the *search version* we would ask for such a coloring. Importantly, an algorithm solving the optimization problem directly solves the decision problem. Indeed, given $m^*(i)$ it is simple to verify if $m^*(i) \leq k$.

Most of optimization problems of practical relevance belong to so-called class NPO:

Definition 2.1.1 *An optimization problem (I, F, m) is in class NPO if it satisfied the following criteria:*

- *the set I is recognizable in polynomial time*
- *feasible solutions have polynomially-bounded size, i.e. there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{R}$ such that for each $y \in F(i)$ we have $|y| \leq p(|i|)$. Moreover, for all y such that $|y| \leq p(|i|)$ we can decide in polynomial time if the solution y is feasible for the instance i .*
- *the measure function m is computable in polynomial time*

2.2 Complexity of optimization problems

The complexity analysis of an optimization problem essentially indicates the scaling of the amount of resources required to solve the problem on *the size of the input instance*. The size of the instance is usually defined as the number of bits that are necessary to encode it. We remark

that different encodings lead to different size evaluations. However, all *natural* encoding schemes that do not introduce unnecessary redundancy are polynomially related [Ausiello *et al.*, 1999].

An algorithm solving an optimization problem defines an upper bound on its runtime complexity. We notice that an algorithm may have very different behavior on instances of the same size. For example, the greedy approach easily finds an optimal coloring for any bipartite graph $G_b = (V_b, E_b)$ while it may have very poor performance on another graph with the same number of nodes and edges. Therefore, the computational complexity considers the runtime scaling in the *worst-case*.

For *decision problem* we say that it is solvable in $O(g(n))$ if there exists an algorithm \mathcal{A} such that for any instance i of size $|i| \leq n$ the algorithm \mathcal{A} decides if the instance is positive in the runtime bounded by $g(n)$.

A *lower bound*, if there is one, indicates how much time is *necessary* for *any* algorithm that finds an optimal solution. Surely, lower bounds are much more difficult to establish than upper bounds. The lower-bound analysis is particularly challenging for problems with an exponential upper bound. Therefore, we typically analyze the complexity of difficult optimization problems in relation to other problems. In other words, we establish the complexity by proving that a problem is at least as difficult as another problem that is believed to be hard. Formally, this relation is captured in the classification of the problems on *complexity classes*.

2.2.1 NP-hardness

The framework of the complexity analysis was historically formulated in the terms of decision problems. As we have seen before, each optimization problem is at least as difficult as its decision version for a fixed k . We remark that the inverse is in general not true. For instance, for a constraint satisfaction problem over m clauses each containing at most 2 binary variables, it is easy to decide if we can simultaneously satisfy all $k = m$ constraints. However, finding an assignment that maximizes the number of satisfied clauses is a much more difficult task [de Beudrap *et al.*, 2021].

NP-completeness of decision problems

The mathematical logic operates the concept of *decidability*. *Decidability* indicates if a decision problem can be solved at all by a computer program. Roughly speaking, a problem is decidable if there exists a program that terminates on every input with a correct positive or negative answer. We remark that by definition such a program may take an arbitrary finite time until termination.

In reality, we are not satisfied with a result that takes too long to be computed. Moreover, arguably all optimization problems of practical relevance are already decidable with a trivial exhaustive search. Therefore, rather than separating problems by their decidability we distinguish *tractable* and *intractable* problems. Conventionally, we consider a problem to be tractable if it can be solved in polynomial time. Tractable problems form the well-known complexity class P.

In quantum computing an analog to P is BQP - a class of problems solvable in polynomial time on quantum computers with an error probability of at most $1/3$ on all instances (positives as negatives). Problems that are solvable with high probability in polynomial time by a probabilistic *classical* algorithm form so-called class BPP. Classes PostBPP and PostBQP correspond to the extensions of BPP and BQP respectively where we allow the algorithms to perform post-selection on the outputs of probabilistic samplings.

We remark that in real-world applications the division on tractable and intractable problems is still too coarse, as computational algorithms scaling as $O(n^d)$ for high d may take weeks even

on an input of a rather modest size.

In a typical situation, a decision problem admits a *constructive solution*. For instance, a proper k -coloring is a constructive solution for the decision version of the graph coloring problem that asks if there exists a k -coloring. A constructive solution may be interpreted as a certificate for $i \in I_P$. A certificate can be used in a procedure that verifies if the constructive solution indeed proves that the instance is positive. If the verification can be done in polynomial time, we say that the problem belongs to the class NP:

Definition 2.2.1 *A decision problem \mathcal{P}_D is in the class NP if there exists a polynomial-time computable function V called **verifier** such that*

- *for each positive instance $i \in I_p$ there exists a certificate $c(i) \in C$ of size $O(\text{poly}(n))$.*
- *Given a instance i and a certificate c the verifier $V : I \times C \rightarrow \{0,1\}$ accepts a valid certificate for a positive instance $i \in I_p$, i.e.*

$$\forall i \in I_p \exists c_i, |c_i| \leq O(\text{poly}(n)) \text{ such that } V(i, c_i) = 1 \quad (2.2)$$

- *For all negative instance $i \in I_n$ and any certificate c the verifier rejects the proof:*

$$\forall i \in I_n \forall c, |c| \leq O(\text{poly}(n)) \text{ we have } V(i, c) = 0 \quad (2.3)$$

We remark that the class NP can be equivalently defined via non-deterministic algorithms. More difficult *counting problems* that ask *the number* of accepting certificates for some $\mathcal{P}_D \in \text{NP}$ (if any) form the class #P.

A quantum version of the NP class is the class QMA. QMA is similarly defined as a class of problems for which positive instances can be certified in polynomial time. The crucial difference with NP is that in QMA the certificate can be a quantum state and the verifier has an access to a quantum computer. In addition, we remark that as the quantum program returns a probabilistic output the conditions (2.2) and (2.3) have to be satisfied with *high probability* rather than *exactly*.

In classical computing, the probabilistic verification of the conditions (2.2) and (2.3) appears in the *Arthur-Merlin complexity class*.

A decision problem $\mathcal{P}_D \in \text{NP}$ is called NP-complete if any other problem \mathcal{P}'_D from the class NP there is a polynomial procedure $f : I \rightarrow I'$ transforming instances of \mathcal{P}'_D to instances of \mathcal{P}_D in a way that preserves positivity:

$$i \in I_p \iff f(i) \in I'_p \quad (2.4)$$

Informally speaking, NP-completeness of \mathcal{P}_D implies that given a polynomial algorithm solving \mathcal{P}_D , we can solve any other problem from NP in polynomial time. The famous Cook-Levin theorem was the first proof of NP-completeness for the *Boolean Satisfiability* problem [Cook, 1971]. Since then many problems were proven to be NP-complete, mainly by reduction to the problems for which NP-completeness was previously established. An extensive list of NP-completeness results may be found in [Garey and Johnson, 1979].

It turns out that among NP-complete problems some are more difficult than others. In particular, this is the case for decision problems that, while being untractable in polynomial time on the size of the instance $|i|$, can however be solved in $O(\text{poly}(|i|, \max(i)))$ where $\max(i)$ is the number of the largest magnitude in the instance. Such problems are called pseudopolynomial. For instance, the decision version of the *Knapsack problem* is a pseudopolynomial problem.

On the opposite side, NP-complete problems that remain intractable even when the runtime is allowed to scale as $O(\text{poly}(|i|, \max(i)))$ are called strongly NP-complete [Garey and Johnson, 1978].

Complexity of optimization problems

An important connection between an optimization problem \mathcal{P} and the corresponding decision problem \mathcal{P}_D is that if $\mathcal{P} \in \text{NPO}$ then $\mathcal{P}_D \in \text{NP}$ [Ausiello *et al.*, 1999]. Optimization problems solvable in polynomial time belong to so-called class PO. The analogue to the *NP-completeness* in the framework of optimization problems is captured by the notion of *NP-hardness*.

Definition 2.2.2 *An optimization problem \mathcal{P} is NP-hard if every decision problem $\mathcal{P}'_D \in \text{NP}$ can be solved in polynomial time given an oracle for \mathcal{P} .*

In other words, the NP-hardness of the optimization problem \mathcal{P} implies that if there were a polynomial algorithm \mathcal{A} that finds an optimal solution $x^*(i)$ for each instance i , then any decision problem in the class NP can be solved in polynomial time. Notably, if the decision version \mathcal{P}_D of \mathcal{P} is NP-complete, then \mathcal{P} is NP-hard.

Many combinatorial optimization problems of practical importance are known to be NP-hard. Therefore, under the conjecture $\text{P} \neq \text{NP}$ they are not expected to be solved by polynomial-time algorithms. However, more careful analysis is needed to evaluate the *practical difficulty* of an NP-hard problem.

Firstly, NP-hardness reflects the worst-case behavior. It may happen that the average runtime performance of an algorithm is satisfactory even if it has an exponential scaling in the worst-case. Moreover, in practical applications a specific structure of real-world instances may render the computation easier.

From the theoretical point of view variations in the difficulty of NP-hard optimization problems are captured by the notions of *pseudopolynomial problems* and *strongly NP-hard problems*. An optimization problem $\mathcal{P} \in \text{NPO}$ is pseudopolynomial if it can be solved by an algorithm polynomial on $|i|$ and $\max(i)$ where $\max(i)$ is the magnitude of the largest number occurring in the instance i . For example, in the Knapsack problem the value $\max(i)$ corresponds to the largest weight or the largest profit of an item in the instance. Notably, instances of a pseudopolynomial problem with polynomially bounded magnitudes $\max(i) \leq O(\text{poly}(|i|))$ can be handled in polynomial time.

Strong NP-hardness for optimization problems is similar in spirit to the *strong NP-completeness* for decision problems. Formally, for each optimization problem \mathcal{P} and a polynomial p we can define a problem $\mathcal{P}^{\max, p}$ that is the restriction of \mathcal{P} on instances such that $\max(i) \leq p(|i|)$. A problem $\mathcal{P} \in \text{NPO}$ is *strongly NP-hard* if there exists a polynomial p such that $\mathcal{P}^{\max, p}$ is NP-hard. For instance, many seminal optimization problems such as MaxCut, Maximum Independent Set, and graph coloring are strongly NP-hard [Garey and Johnson, 1978].

As follows from the definition, strongly NP-hard problems cannot be pseudopolynomial.

Finally, in the context when the exact solution is unreachable in reasonable time we may instead consider *near-optimal solutions*. A near-optimal solution is usually referred to as *approximate solution* or *approximate optimum*. The complexity to reach an approximate solution of certain quality leads to a new classification of NP-hard optimization problems in terms of their approximability.

2.3 Approximation algorithms

NP-hardness usually implies that it is impossible to compute the exact optimal solution for the optimization problem on instances of relevant sizes in an acceptable time. A way to address

this inconvenience is to consider near-optimal solutions that can be potentially easier to find. Polynomial-time algorithms returning near-optimal solutions of NP-hard optimization problems are called approximation algorithms. Approximation algorithms can be divided into *algorithms with guaranteed performance* and *heuristics*.

2.3.1 Approximation ratio

We denote by \mathcal{A} an approximation algorithm for an optimization problem $\mathcal{P} = (I, F, m)$. We distinguish several types of errors that are due to the non-optimality of the solution returned by \mathcal{A} :

- *absolute error*. For an instance i with optimal solution $m^*(i)$ the absolute error corresponds to the difference between the optimal solution and the solution returned by the approximation algorithm:

$$E(i) = |m^*(i) - m(i, \mathcal{A}(i))| \quad (2.5)$$

- *relative error*. This metric captures a relative deviation from the optimum value:

$$r(i) = \frac{|m^*(i) - m(i, \mathcal{A}(i))|}{\max\{m^*(i), m(i, \mathcal{A}(i))\}} \quad (2.6)$$

In the current thesis we extensively use another performance metric called *approximation ratio*. For a minimization problem the approximation ratio of an algorithm \mathcal{A} on an instance $i \in I$ is $\alpha(i) = \frac{m^*(i)}{m(i, \mathcal{A}(i))}$. Alternatively, for the maximization problem the approximation ratio is $\alpha(i) = \frac{m(i, \mathcal{A}(i))}{m^*(i)}$. We remark that the approximation ratio is always lower or equal to one. In the case of equality $\alpha(i) = 1$ we can safely claim that the approximation algorithm found an optimal solution.

In the formal analysis of the performance of approximation algorithms we are typically interested in the worst-case behavior. We call an algorithm an α -approximate algorithm if it has the approximation ratio of at least α on *any* instance of the optimization problem, i.e.

$$\forall i \in I : \alpha(i) \geq \alpha \quad (2.7)$$

We say that the bound 2.7 is tight if there exists at least one instance $i^* \in I$ such that the approximation ratio $\alpha(i^*)$ is precisely α .

An optimization problem \mathcal{P} is called α -approximable if there exists a polynomial-time α -approximate algorithm solving it.

NP-hard problems have different approximability properties. Some can be approximated to any desired error rate while for others the relative quality of the best possible algorithm is always poor. In the latter case, the approximation ratio usually decreases with the size of the instance.

In this section, we consider classes of problems with different achievable approximation guarantees.

2.3.2 APX

Definition 2.3.1 *APX is a class of optimization problems such that there exists a polynomial-time approximate algorithm with approximation ratio α bounded by a constant $\alpha_1 \leq \alpha$.*

For instance, the MaxCut problem belongs to the class APX [Goemans and Williamson, 1995].

To show that an optimization problem is in APX it suffices to provide an algorithm that has a constant approximation ratio α_l . Usually, we are interested in known if the achieved ratio is tight, i.e. if there can be a (probably still undiscovered) algorithm that has a better performance. For some problems it can be shown that if it were possible to approximate the solution with a factor $\alpha > \alpha_u$, then $P = NP$. If the approximation ratio of the algorithms is $\alpha_l = \alpha_u$ then in a sense it provides the best achievable approximation performance.

However, for many problems the strong inapproximability results leave a gap $\alpha_u - \alpha_l > 0$ with the best-known algorithms. If this is the case, we can check if a stronger bound $\alpha < \alpha'_u < \alpha_u$ can be derived assuming complexity conjectures that are weaker than $P \neq NP$. Such inapproximability result was derived, for instance for the MaxCut problem assuming the *Unique Game Conjecture (UGC)* [Khot *et al.*, 2007]. The UGC-bound for the MaxCut implies, in particular, the tightness of the Goemans-Williamson approximation ratio $\alpha = 0.878 \dots$ [Khot *et al.*, 2007].

In general, there exist problems in NPO that are even not in APX. In other words, for such problems there is no algorithm returning a solution of any guaranteed constant ratio unless $P = NP$. A seminal example of a problem $\mathcal{P} \notin \text{APX}$ is the *Maximum Independent Set* problem [Håstad, 1999].

The proofs $\mathcal{P} \notin \text{APX}$ can be derived using various techniques. For instance, we can show that there exists an NP-complete decision problem that can be reduced to the approximation problem with arbitrary fixed ratio α . Such approach is called *gap technique*. An illustration of the gap technique for the inapproximability bound of the Traveling Salesperson problem can be found in [Ausiello *et al.*, 1999]. More involved proofs are usually derived from a fundamental result of the complexity theory called *PCP-theorem* [Arora and Safra, 1998].

2.3.3 PTAS

There exist optimization problems for which an exact solution is difficult to compute but it can nevertheless be approximated to any desired ratio. Formally speaking, for such a problem there exists a family of α -approximation algorithms for all $\alpha < 1$. Such a family is called *polynomial-time approximation scheme* or *PTAS* for short. A slightly weaker notion is that of *asymptotic approximation scheme* or, equivalently, a family of polynomial-time algorithms such that the approximation ratios $\alpha < 1$ are achieved in the limit of large instances.

We remark that usually an approximation scheme takes more time to find solutions for better approximation ratios. In fact, for a polynomial-time approximation scheme, the dependence on the desired solution quality can still be exponential. This is the case, for instance, for the PTAS solving the smart scheduling problem considered in the current thesis [Skutella and Woeginger, 2000].

We say that an optimization problem $\mathcal{P} \in \text{NPO}$ belongs to the class PTAS if it admits a polynomial-time approximation scheme. An example of a problem in PTAS is the *Maximum Independent Set (MIS)* in a planar graph [Ausiello *et al.*, 1999]. We remark that the MIS problem on non-planar graphs is much harder, to such an extent that it is not even in APX.

Importantly, if $P \neq NP$ the class PTAS is strictly contained in APX, i.e. $\text{PTAS} \subsetneq \text{APX}$.

2.3.4 FPTAS

The runtime of PTAS can be significantly increased for more demanding performance guarantees. However, for some problems there exists a family of α -approximation algorithms with the

complexity that scales polynomially *both* on the size of the instance $|i|$ and on the desired ratio α . Such family is called a *fully-polynomial approximation scheme (FPTAS)*. For an NP-hard optimization problem this is in a sense the strongest possible approximability result. Roughly speaking, the existence of FPTAS for $\mathcal{P} \in \text{NPO}$ implies that even if the exact solution is difficult to compute, it can be efficiently approximated to any desired ratio. The class of the optimization problem that admits such a scheme is denoted as FPTAS. An example of the problem from FPTAS is the Maximum Knapsack problem.

The book [Garey and Johnson, 1978] identifies certain conditions on optimization problems that guarantee that the problem does *not* possess an FPTAS. For instance, this is the case for *polynomially bounded* strongly NP-hard problems. We recall that a problem is called polynomially bounded if its measure function is bounded, i.e. $\forall i \in I$ and $\forall y \in F(i) : m(x, y) \leq p(|x|)$ for some polynomial p .

On the opposite sense, an existence of FPTAS for a problem \mathcal{P} such that $m^*(x) < p(|x|, \max(x))$ implies that \mathcal{P} is pseudo-polynomial.

In most cases an FPTAS for a problem \mathcal{P} , if it exists, is derived from *dynamic programming* formulation. For instance, if the dynamic programming formulation has a special property introduced in [Woeginger, 2001] under the name *DP-benevolence*, then the problem admits an FPTAS. DP-benevolence is a particularly interesting sufficient condition for the existence of FPTAS as *i)* it is easy to verify *ii)* it captures a relatively large family of optimization problems. Typically, FPTAS is derived from a dynamic programming formulation with *rounding-the-input-data* technique [Sahni, 1976]. Another approach originally introduced in [Ibarra and Kim, 1975] is called *trimming-the-state-space* technique.

We remark that the *smart scheduling* problem considered in chapter 4 admits an FPTAS.

2.3.5 Heuristics

Approximation algorithms with guaranteed ratios are crucial in the evaluation of the theoretical complexity of optimization problems. In practice, however, they are often outperformed by *heuristics*. Heuristics are polynomial-time programs without performance guarantees that, nevertheless, demonstrate good behavior in experimental evaluations on instances of practical interest. On the other side, heuristics may have very poor performance in the worst case. Some works include in the definition of heuristics methods with an exponential scaling in the worst case if they are relatively rapid on relevant instances.

Heuristics are usually inspired by one of two traditional approaches. The first one consists of an iterative construction of the solution from a partial solution. For instance, greedy algorithms implement such a *constructive approach*.

Alternatively, *local search heuristics* start from a *complete feasible solution* and try to improve it. For this purpose, they search for an improving solution inside the *neighborhood* of the actual solution. Contrary to greedy algorithms that are usually tailored for a specific application, local search is a very general framework. It is implemented by various metaheuristics such as simulated annealing [Kirkpatrick *et al.*, 1983] and tabu search [Battiti and Protasi, 2001, Kochenberger *et al.*, 2013]. The performance of local search heuristics heavily depends on the quality of the initialization point. Usually, the initialization point is found by a constructive heuristic. In addition, local-search metaheuristics have a large number of parameters that should be efficiently fine-tuned to assure good practical performance.

In the last decades emerged heuristics inspired by the *natural selection* process. For instance, an imitation of natural selection was realized in genetic algorithms [Marchiori, 1998] and the differential evolution algorithm.

The domain of heuristic algorithms is large and continuously developing. A comprehensive overview of the most important heuristic methods for combinatorial optimization can be found in [Burke and Kendall, 2006].

2.4 Semidefinite programming

Semidefinite programming is a subfield of *convex optimization* over *continuous variables*. In the field of combinatorial optimization semidefinite programs were found to be extremely useful in the design of approximation algorithms with guaranteed approximation ratios. We refer to [Gärtner and Matoušek, 2013] for a general introduction to the field.

A semidefinite program is a problem of maximizing a linear function over n^2 continuous variables $x_{i,j}$. Variables may be arranged in a matrix $X = \begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ & \ddots & \\ x_{n,1} & \dots & x_{n,n} \end{pmatrix}$.

Semidefinite program (SDP) is formulated as follows:

$$\min \operatorname{Tr}(A^T X) \tag{2.8}$$

$$\operatorname{Tr}(C_k^T X) \leq b_k, \quad k \in [1, \dots, m] \tag{2.9}$$

$$X \succeq 0 \tag{2.10}$$

where $\operatorname{Tr}(Y)$ denotes the trace of the matrix Y and $X \succeq 0$ means that the matrix X is positive semidefinite. We remark that the objective function and the constraints are linear functions on variables $x_{i,j}$ as $\operatorname{Tr}(A^T X) = \sum_{i,j} a_{i,j} x_{i,j}$.

Under certain relatively mild conditions, SDP can be solved up to any desired accuracy in polynomial time. In particular, *trace-bounded instances* can be efficiently approximated with an algorithm implementing multiplicative weight update [Arora and Kale, 2016].

Goemans-Williamson algorithm

Many optimal approximation algorithms for combinatorial problems are based on semidefinite programming. Usually, an SDP is formulated as a continuous relaxation of a problem over integer variables. For instance, this is the case for the *Goemans-Williamson algorithm* for the *MaxCut* problem [Goemans and Williamson, 1995].

MaxCut problem on a graph $G = (V, E)$ is a search of a partition $V = V_1 \sqcup V_2$ such that the number of crossing edges $(u, v) : u \in V_1, v \in V_2$ is maximized. We assign one variable $z_v \in \{-1, 1\}$ to each node. The partition corresponding to a variable assignment $\mathbf{z} \in \{-1, 1\}^n$ is exactly $V_1 = \{v \mid z_v = 1\}$ and $V_2 = \{v \mid z_v = -1\}$. The MaxCut problem may be written as:

$$\max \sum_{(u,v) \in E} \frac{1 - z_u z_v}{2} \tag{2.11}$$

The Goemans-Williamson algorithm is based on a semidefinite relaxation of the MaxCut problem. The value of the relaxation provides an upper bound on the size of the optimal cut $c^*(G)$.

In the Goemans-Williamson algorithm we associate to each node a vector y_v on the unit sphere \mathbb{S}^n in \mathbb{R}^n , i.e. $\mathbb{S}^n = \{x \in \mathbb{R}^n \mid |x| = 1\}$.

With vectors from the unit sphere, a "continuous" version of the MaxCut can be written as:

$$\max \sum_{(u,v) \in E} \frac{1 - y_u^T y_v}{2} \quad (2.12)$$

$$y_v \in S^n, \quad \forall v \in V \quad (2.13)$$

We remark that the formulation over $y_v \in S^n$ is the relaxation of the integer problem 2.11. Indeed, an optimal solution for the integer problem leads to a feasible solution $y_v = [0, \dots, 0_{v-1}, z_v, 0_{v+1}, \dots, 0]$ for the vector problem with the same objective value. Therefore, the optimum solution for (2.12) has the value at least $c^*(G)$.

From the formulation over y_v we can get an equivalent SDP program. We denote by $x_{u,v} = y_u^T y_v$. An equivalent SDP program is:

$$\max \sum_{(u,v) \in E} \frac{1 - x_{u,v}}{2} \quad (2.14)$$

$$x_{u,u} = 1, \quad u \in [1, \dots, n] \quad (2.15)$$

$$X \succeq 0 \quad (2.16)$$

In the SDP formulation the condition $y_v \in S^n$ was translated to $x_{u,u} = 1$. Proof of equivalence of both problems can be found in [Gärtner and Matoušek, 2013]. We remark that this proof is constructive, i.e. it explicitly specifies how to recover the vectors y_v from the matrix X . Therefore, an optimal solution X^* of the SDP relaxation associates to each node a vector y_v^* from the unit sphere S^n .

An integer solution for the MaxCut is obtained from vectors y_v^* with a help of a *random rounding plane*. To be exact, the algorithm chose a vector $\rho \in \mathbb{R}^n$ at random. The integer solution is recovered from the separation of \mathbb{R}^n on two subspaces generated by ρ :

$$z_v^* = \begin{cases} 1, & \text{if } \rho^T y_v^* \geq 0 \\ -1, & \text{otherwise} \end{cases} \quad (2.17)$$

It was proven in [Goemans and Williamson, 1995] that such an algorithm achieves an approximation ratio of $0.878\dots$. Moreover, this ratio is *tight* [Karloff, 1999] and *optimal* under the Unique Game Conjecture [Khot *et al.*, 2007].

Part II

Quantum algorithms for Smart Charging problems

Chapter 3

Quantum algorithms for combinatorial optimization

3.1 Motivation

A lot of combinatorial optimization problems such as *3-SAT*, *Maximum Cut*, *Maximum Independent Set*, *Graph Coloring*, *Traveling Salesman*, *Integer Linear Program* and many others are known to be NP-hard. Therefore, unless the widely believed conjecture $P \neq NP$ is false, these problems don't admit polynomial-time algorithms solving them to optimality [Garey and Johnson, 1979]. There exists, however, exact algorithms that explore the solution space either *explicitly* (a brute-force approach) or *implicitly* by using relaxations that in some cases allow to efficiently eliminate suboptimal solutions (Branch & Bound). Obviously, exact algorithms have an exponential runtime in the worst-case.

Another way to practically address NP-hard problems consists in relaxing the requirement of optimality. Indeed, in a restricted-time regime, it is preferable to have a suboptimal solution (*approximate optimum*) rather than wait too long for the exact optimum to be computed. Two types of polynomial algorithms implement this idea: approximation algorithms and heuristics [Ausiello *et al.*, 1999]. While the approximation algorithms provide theoretical guarantees on the quality of the obtained solution, the heuristics are usually validated by experimental demonstrations. In this setting, the performance is determined by the resource consumption (*runtime and memory*) but also by the quality of the solution (*the approximation ratio*). The approximation ratio indicates how far the approximate solution is from the exact optimum.

Hence the question arises if quantum computers can provide an advantage for combinatorial optimization applications, either by *speeding up* the execution of exact or approximate routines or via new heuristics that improve the *quality* of the approximate solution over state-of-the-art classical methods either in theory or in practice.

3.1.1 Quantum acceleration of classical routines

Indeed, the integration of quantum routines can *accelerate* both approximate and exact optimization algorithms.

The exact optimization typically profits from Grover's algorithms for the search in an unstructured database [Grover, 1996]. Grover's original algorithm assures a quadratic speedup over the best-possible classical counterpart [Grover, 1996]. This technique can provide a polynomial speedup for routines solving the SAT problem [Ambainis, 2005], for the Branch & Bound

[Montanaro, 2020] or for the *nested search* for structured problems [Cerf *et al.*, 2000].

Furthermore, a quantum algorithm with a quadratic speedup was suggested for the *semidefinite programming problem (SDP)* [van Apeldoorn *et al.*, 2020]. SDP followed by a randomized rounding often appears as a subroutine in the approximate algorithms for binary optimization with the best-established performance guarantees [Gärtner and Matoušek, 2013]. The quantum speedup for SDP implies a proportional acceleration for these approximation routines for such problems as MaxCut [G.S L. Brandão *et al.*, 2022].

However, even if the asymptotic time complexity of the algorithms mentioned above improves over their classical counterparts the exponential scaling is preserved and no significant improvement over a random guess is expected in a constrained time budget [McClean *et al.*, 2021]. Moreover, for asymptotic polynomial speedups a practical runtime advantage is not expected to be observed on the NISQ machines [Babbush *et al.*, 2021]. This limitation is due to heavy error-correction schemes that are necessary to ensure the correctness of the circuit execution. Error correction seems to be the genuine bottleneck for practical quantum computing: even for quantum heuristics (believed to be NISQ-compatible) fault-tolerant realizations induce a significant resource consumption [Sanders *et al.*, 2020].

3.1.2 Quantum heuristics

On the other hand, quantum heuristics can potentially improve over the classical ones in terms of the *quality of the approximate solution*. In addition, these heuristics (among them *Quantum Annealing (QA)*, *Quantum Approximate Optimization Algorithm (QAOA)* [Farhi *et al.*, 2014a] and *Variational Quantum Eigensolver (VQE)* [Peruzzo *et al.*, 2014]) are expected to be well-suited for the NISQ devices [Preskill, 2018]. Indeed, these algorithms can be implemented in a low-depth regime and because of their approximate nature, one can bypass the explicit error correction.

Unfortunately, the actual error rate leaves serious doubts on the ability of quantum heuristics to demonstrate quantum advantage [Albash *et al.*, 2017, França and García-Patrón, 2021, Lotshaw *et al.*, 2022]. The limiting result [França and García-Patrón, 2021] is due to the existence of a thermal Gibbs state $G_T = e^{-H/T} / \text{Tr}(e^{-H})$ that has the same mean energy $\langle H \rangle$ as the quantum state returned by a noisy implementation of a quantum heuristic. The corresponding temperature T of the state depends on the noise level. Crucially, for sufficiently high noise, the temperature T is high enough to allow an efficient classical sampling from G_T . Therefore, the quantum advantage is unlikely *unless* at least one of the two conditions is fulfilled: the problem matches the hardware-native architecture or the noise level is significantly reduced (by two orders of magnitude from the actual values).

This chapter introduces quantum algorithms for the combinatorial optimization that are believed to be adapted for NISQ: *Quantum Annealing (QA)*, *Quantum Approximate Optimization Algorithm (QAOA)* and *Recursive QAOA (RQAOA)*. We concentrate on the analysis in the noise-free regime and only briefly mention the results about the impact of decoherence on physical realizations.

Ground state problem

Traditionally, a quantum combinatorial optimization algorithm receives as input an observable H called Ising Hamiltonian. From the physical side, Ising Hamiltonians are used to describe the energy levels of quantum many-body systems with pair interactions such as spin-glass models [Zarinelli, 2012]. Mathematically, an Ising Hamiltonian is a Hermitian operator diagonal in

computational basis that can be expressed as a polynomial of degree 2. The original framework was further expended to deal with higher-degree interactions [Hadfield, 2018, Glos *et al.*, 2022] and to incorporate hard restrictions on the solution space [Hadfield *et al.*, 2019, Hadfield, 2018].

The problem we are interested in consists in finding the ground state (eigenstate with the smallest eigenvalue) for a given Hamiltonian. Quantum heuristics will typically return not exactly the *ground state* but a *low-energy state* that is considered as an approximate solution.

In order to apply quantum algorithms to combinatorial optimization problems we first need to map the solution space S to a subset S_b of computational basis states of an n -qubit system $|x\rangle$, $x \in \{0, 1\}^n$. Then we need to define a corresponding diagonal Hamiltonian H such that for every feasible solution its eigenvalue corresponds to the value of the objective function:

$$H|x\rangle = f(x)|x\rangle, \quad |x\rangle \in S_b \quad (3.1)$$

The translation is straightforward for *Quantum Binary Optimization Problems (QUBO)* and relatively easy for a lot of NP-hard optimization problems [Lucas, 2014]. Section 3.2 contains a detailed description of the Ising Hamiltonian ground state problem and related complexity results.

Quantum evolution in algorithms for optimization

In section 3.3 we start the exploration of quantum optimization algorithms with the **Adiabatic Algorithm (AA)** originally introduced in [Farhi *et al.*, 2000]. By definition, the Adiabatic Algorithm returns an exact solution for a given optimization problem. The AA initializes the system in an easy-to-prepare ground state of some Hamiltonian H_{init} . The state evolves under the time-dependent Hamiltonian which gradually changes from H_{init} to the H_{final} where the ground state of H_{final} corresponds to the solution of the optimization problem.

According to Schrödinger's equation, the time evolution of any quantum system is governed by the Hamiltonian acting on it:

$$i\hbar \frac{\partial_t |\psi(t)\rangle}{\partial t} = H(t) |\psi(t)\rangle \quad (3.2)$$

where $\hbar = 1.054 \cdots \times 10^{-34} Js$ is the *reduced Plank constant*.

The *adiabatic theorem* states that if the evolution is sufficiently slow (with respect to the difference between the ground state energy and the energy of the first excited state usually called *spectral gap*), the system remains close to the instantaneous ground state during the whole process [Jansen *et al.*, 2007, Amin, 2009a]. Therefore, with high probability the measure output of the final state will be the ground state of H_{final} - precisely the exact solution of the optimization problem. The downside, although consistent with the conjecture $NP \not\subseteq BQP$, is that "sufficiently slow" is generally difficult to compute and sometimes the best bounds imply an exponential evolution runtime for difficult instances of NP-hard problems [van Dam *et al.*, 2001] or even for some classically tractable problems [Reichardt, 2004, Albash and Lidar, 2018]. Even worse scaling when the gap vanishes faster than exponentially was observed for random instances of the NP-hard Exact 3 Cover problem [Altshuler *et al.*, 2010].

In addition, even in the case of a polynomial gap, the adiabatic runtime condition is difficult to satisfy on machines with a realistic coherence time limit. Therefore the Adiabatic Algorithm remains mostly a theoretical construction. It has, however, inspired several polynomial-time heuristics such as Quantum Annealing and QAOA [Farhi *et al.*, 2014a].

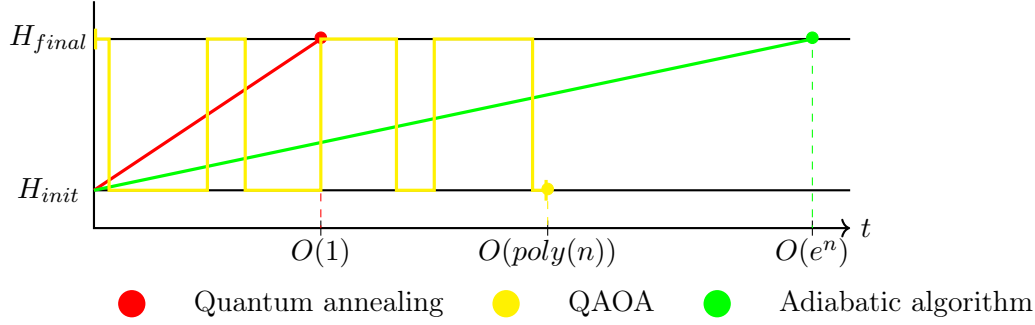


Figure 3.1: Hamiltonians governing the evolution of the quantum system at different moments of the execution of optimization algorithms.

The **Quantum Annealing** changes the Hamiltonian in a similar way that the Adiabatic Algorithm but much faster than allowed by the adiabatic condition (see figure 3.1). The continuous evolution is performed on special-purpose devices called quantum annealers, and the runtime is typically limited by the hardware characteristics. In the *DWave Advantage 6.1* the available annealing time ranges from $0.5\mu s$ to $2000\mu s$ [Systems, 2022]. The Quantum Annealing returns a solution with no guarantees on its quality and an experimental benchmark is required to conclude about the potential of this approach. An experimental study reported in [King *et al.*, 2015] claims that, under a fixed time budget, Quantum Annealing outperforms the state-of-the-art classical solvers on some optimization problems. On the other hand, the work [Martin-Mayor and Hen, 2015] demonstrates that QA scales worse with the *hardness* of the instance than the thermal classical algorithm. In section 3.4 we report some experimental studies concerning the application of Quantum Annealing to optimization problems.

In contrast to the Quantum Annealing that requires a special-purpose analog device, the **Quantum Approximate Optimization Algorithm** (see section 3.5) adapts the ideas of the Adiabatic Algorithm to the universal gate-based quantum computers [Farhi *et al.*, 2014a]. Beyond the usual considerations about the relative power of two types of hardware⁵ [Barends *et al.*, 2016] the gate-based computers, contrary to existing annealers that allow only pair interactions, can directly deal with polynomials of higher degrees [Glos *et al.*, 2022]. This property is useful in practice. For instance, the E3LIN2 problem for which the quantum advantage was firstly obtained and then beaten by a new classical algorithm, has cubic terms [Farhi *et al.*, 2014b]. In addition, the ability to directly treat higher degree interactions sometimes leads to implementations with lower resource complexity [Glos *et al.*, 2022, Fuchs *et al.*, 2020].

The QAOA dynamics is ruled not by a continuous interpolation of two Hamiltonians but by a sequence of pulses ("bangs") that jumps between H_{init} and H_{final} (see figure 3.1). The total amount of pulses $p \in \mathbb{N}$, called *depth*, is predefined by the user while the durations of bangs are optimized directly inside the algorithm. The depth of the quantum circuit that performs the evolution scales proportionally with the QAOA depth, so due to the limited coherence time the near-future implementations are likely to stay in the regime $p = O(1)$.

Hence, the runtime of QAOA is determined by two factors: the depth p and the total sum of pulse durations. It was observed that if the latter scales as the inverse polynomial on the problem's size then no quantum advantage is possible [Hadfield *et al.*, 2021].

⁵In theory, Adiabatic evolution under general Hamiltonian is universal [Aharonov *et al.*, 2007]

Which quantum heuristic is better?

Both Quantum Annealing and QAOA alternate between H_{init} and H_{final} :

$$H(t) = (1 - g(t))H_{init} + g(t)H_{final}$$

where for the annealing the function $g(t)$ is continuous and for QAOA $g(t)$ has the pulse form. The work [Yang *et al.*, 2017] suggested that for a fixed time limit a "bang-bang" alternation protocol (as in QAOA) is optimal in terms of the achievable approximation ratio. More recently, another study reported that one of the assumptions used to prove the "bang-bang" optimality doesn't hold in general case [Brady *et al.*, 2021]. This study shows that for a fixed evolution time, the *bang-anneal-bang* sometimes is better, in particular for some instances encoded by Ising Hamiltonians. However, in a more realistic regime where the depth is given but the sum of pulse durations is not fixed QAOA's performance rapidly approaches the one of the bang-anneal-bang protocol [Brady *et al.*, 2021]. Moreover, QAOA has a practical advantage: the pulse control is expected to be easier to optimize and execute than a bang-anneal-bang protocol.

The deep connection between QAOA and Quantum Annealing is also reflected by the fact that the discretized annealing schedule, when used to initialize the search of QAOA pulse durations, generally leads to better QAOA schedules compared to the random initialization [Sack and Serbyn, 2021].

Recently, there has been remarkable progress in the physical implementation of annealers: 5000 qubits are accessible on *DWave Advantage system 6.1* [Systems, 2022] compared to relatively modest 127 qubits on the universal *IBM Eagle* quantum computer [Dial, 2022]. Hence, QA is more amenable to experimental analysis. At the same time, due to the lack of tools, the theoretical analysis of the QA is significantly harder and only non-tight guarantee bounds [Braida *et al.*, 2022] and no-go results were established [Moosavian *et al.*, 2022].

On the other side, for QAOA a lot of theoretical results were obtained. Between these results, the particularly important ones are *supremacy* [Farhi and Harrow, 2016] and *universality* [Lloyd, 2018]. Some tight performance guarantees for special instances of MaxCut [Farhi *et al.*, 2014a, Hadfield, 2018, Wurtz and Love, 2021, Marwaha, 2021, Basso *et al.*, 2022] and E3LIN2 [Farhi *et al.*, 2014b] were established, although no quantum advantage was proven. The works [Bravyi *et al.*, 2020, Farhi *et al.*, 2020, Chou *et al.*, 2021] report limiting results on the approximation ratio for constant-depth QAOA on MaxCut and Maximum Independent Set problems. Finally, a general analytical framework for QAOA was suggested in [Hadfield *et al.*, 2021].

Both QA and QAOA are expected to approach the exact optimum at the infinite time limit, but the convergence speed is unknown. Several results were established for QAOA that demonstrate that at least a logarithmic depth is required in order to outperform the classical worst-case approximation ratio guarantees for MaxCut [Bravyi *et al.*, 2020], Max-k-XORSAT [Chou *et al.*, 2021] and MIS [Farhi *et al.*, 2020]. Most of these results were then generalized to the short-time Quantum Annealing using *Lieb-Robinson bound* [Moosavian *et al.*, 2022]. A question if a bounded-depth QAOA can show an advantage over local classical algorithms is studied in [Hastings, 2019, Barak and Marwaha, 2022].

The last algorithm presented in this chapter is Recursive QAOA (RQAOA) (see section 3.6). RQAOA was originally designed to deal with *locality* that was often mentioned as the limiting condition for the QAOA performance [Bravyi *et al.*, 2020, Farhi *et al.*, 2020, Hastings, 2019, Chou *et al.*, 2021]. RQAOA proceeds by a recursive elimination of variables, where at each step the variable to eliminate is selected using the optimal QAOA state for the restricted problem. The promise behind this is that, contrary to constant-depth QAOA, RQAOA ends up seeing the *global connectivity structure*. As a demonstration, the work [Bravyi *et al.*, 2020] shows that at

depth $p = 1$ RQAOA finds the optimal solution on 1D Ising chains given by the Hamiltonian $H = \sum_{i=1}^n h_i Z_i Z_{i+1}$, while QAOA's performance is bounded for any fixed p [Mbeng *et al.*, 2019a].

From the experimental point of view, a particularly interesting fact about RQAOA is that for $p = 1$ it can be efficiently simulated on a classical computer [Bravyi *et al.*, 2020, Egger *et al.*, 2021].

In brief, quantum algorithms for combinatorial optimization can be classified by their runtime and the achieved approximation ratio. Thanks to the adiabatic theorem, Adiabatic Algorithm is guaranteed to find an exact solution but potentially in exponential time. QAOA and RQAOA by construction scale polynomially and, while there exist bounds on the approximation ratio in special cases, their performance is mostly evaluated experimentally. In actual implementations Quantum Annealing has a constant runtime and almost only experimental performance demonstrations.

3.2 Ising Hamiltonian

Ising Hamiltonian was originally introduced to model spin interactions in crystals. It was actively used to study such important phenomena as *phase transitions* or the dependency of thermodynamic functions on different parameters [Ising, 1925, Wikipedia, 2022a].

2-local Hamiltonians

As most physical interactions are pairwise, quantum computes are expected to natively support only 2-local connections [Rieffel *et al.*, 2014]. The Ising model is a restriction of 2-local Hamiltonians. A general 2-local Hamiltonian can be written as

$$H = \sum_{(u,v) \in E} [J_{u,v}^x X_u X_v + J_{u,v}^y Y_u Y_v + J_{u,v}^z Z_u Z_v] + \sum_{u \in V} [h_u^x X_u + h_u^y Y_u + h_u^z Z_u] \quad (3.3)$$

where E defines the set of non-zero interactions and X , Y and Z are Pauli matrices. Negative coefficients $J_{u,v} < 0$ correspond to ferromagnetic couplings and positive ones - to antiferromagnetic. If the graph E allows only near-neighbor interaction in k -dimensional lattice the corresponding Hamiltonians are called kD .

The ground state decision problem for such a Hamiltonian is known to be a QMA-complete [Kempe *et al.*, 2006]. QMA is a quantum analog of NP, and QMA-completeness is in a sense the strongest hardness result achievable for decidable problems in quantum computing.

Several special cases may be defined by imposing restrictions on the coefficients J and h . We list below the most famous examples.

- *Heisenberg model* where $J_{u,v}^x = J_{u,v}^y = J_{u,v}^z = J_{u,v}$:

$$H_{\text{Heisenberg}} = \sum_{(u,v) \in E} J_{u,v} (X_u X_v + Y_u Y_v + Z_u Z_v) \quad (3.4)$$

- *XY model* where $J_{u,v}^x = J_{u,v}^y = J_{u,v}$ and $J_{u,v}^z = 0$:

$$H_{XY} = \sum_{(u,v) \in E} J_{u,v} (X_u X_v + Y_u Y_v) \quad (3.5)$$

- *Ising model* where all coefficients are set to zero except h_u^z and $J_{u,v}^z$:

$$H_I = \sum_{(u,v) \in E} J_{u,v} Z_u Z_v + \sum_{u \in V} h_u^z Z_u \quad (3.6)$$

A typical example of an Ising Hamiltonian is the Sherrington-Kirkpatrick mean-field model for a spin glass $H = \frac{1}{\sqrt{N}} \sum_{(u,v) \in G} J_{u,v} Z_u Z_v$ where the coefficients $J_{u,v} \sim \mathcal{N}(0, 1)$ are independent identically distributed standard Gaussian random variables [Sherrington and Kirkpatrick, 1975].

Many combinatorial optimization problems may be encoded in a ground-state search of an Ising Hamiltonian [Lucas, 2014]. Therefore, the problem is NP-hard and exact algorithms (such as Branch & Cut [Simone *et al.*, 1995]) take an exponential time to find the solution.

Transverse-field Ising Hamiltonian

A transverse-field Ising Hamiltonian:

$$H = \sum_{(u,v) \in E} J_{u,v} Z_u Z_v + \sum_{u \in V} h_u^z Z_u + \sum_{u \in V} \sigma_u X_u \quad (3.7)$$

has supplementary mixing component $H_\sigma = \sum_{u \in V} \sigma_u X_u$. The original Ising model is diagonal in the computational basis: the evolution $|x\rangle \rightarrow e^{itH_I} |x\rangle$ under H_I preserves the probability distribution over basis states $|x\rangle, x \in \{0, 1\}^n$. Therefore, the *mixing component* H_σ is essential in quantum algorithms for optimization in order to "balance" the probability weight to better solutions. Mixing is necessary to explore the solution space in a search of an optimal solution. On the other hand, the physical interpretation of mixing term as the *kinetic energy* allows us to demonstrate that sometimes it can drive the evolution *out* of the optimum [McClean *et al.*, 2021].

In some special cases, polynomial algorithms are known for ground-state search of transverse-field Ising Hamiltonians. For instance, a fully-polynomial time approximation scheme for a Hamiltonian with fully ferromagnetic interactions is given in [Bravyi and Gosset, 2017]. Another simple case is that of 1D transverse-field Ising model with $h_u^z = 0$ [Pfeuty, 1970].

An interesting question is the *quantum complexity* of the ground-state search problem for the transverse-field Ising model. Notably, such Hamiltonians are *stoquastic*, i.e. their off-diagonal elements are non-positive in the computational basis [Albash and Lidar, 2018]. Stoquastic Hamiltonian ground state problem belongs to classical *Artur-Merlin complexity class* and, contrary to the case of general 2-local Hamiltonians, it is not believed to be QMA-complete [Bravyi *et al.*, 2006].

From the practical point of view it is interesting to know if there exist instances such that quantum computer can efficiently find their ground states while the best-known classical algorithm takes exponential time. For the moment, only *limited quantum speedup* was observed: the work [Hastings and Freedman, 2013] presents instances that are polynomially solvable on quantum computers but for which some special classical methods like *Simulated Annealing* take exponential time.

3.2.1 Quadratic Unconstrained Binary Optimization (QUBO)

A combinatorial optimization problem that is naturally equivalent to the ground state search of an Ising Hamiltonian is *Quadratic Unconstrained Optimization Problem (QUBO)*:

$$\min_{x \in \{0,1\}^n} x^T Q x \quad (3.8)$$

where Q is a symmetric $n \times n$ matrix. The connectivity pattern of the function generates a graph $G = (V, E)$ where each node $G = (V, E)$ corresponds to a vertex and each edge corresponds to a quadratic term with non-zero coefficient.

The objective goal can be freely changed to maximization by multiplying the function by -1 .

A lot of combinatorial optimization problems such as various forms of assignment problems, sequencing, and ordering [Rieffel *et al.*, 2014, Alidaee *et al.*, 1994], MaxCut [Kochenberger *et al.*, 2013], set covering [Alidaee *et al.*, 2008] and many others may be represented in the QUBO form (often in multiple ways). Moreover, it was shown that *any* unconstrained binary optimization problem can be reduced to QUBO [Anthony *et al.*, 2017]. A list of applications and solution methods may be found in the relatively recent survey [Kochenberger *et al.*, 2014].

The curious thing about QUBO is that sometimes generic QUBO-solvers (such as *Tabu search*) rivals the performance of the specially-designed algorithms for different applications, for example graph coloring [Kochenberger *et al.*, 2005], MaxCut [Kochenberger *et al.*, 2013] and set packing [Alidaee *et al.*, 2008].

For each QUBO the corresponding Ising Hamiltonian H may be found by replacing each binary variable x_i by an operator $\frac{I-Z_i}{2}$ where I is identity and Z is the Pauli-Z operator:

$$Z|x\rangle = (-1)^x|x\rangle, \quad x \in \{0, 1\} \quad (3.9)$$

Indeed, for a vector $|x\rangle$, $x \in \{0, 1\}^n$ from the computational basis we have $\frac{I-Z_i}{2}|x\rangle = \frac{1}{2}(1 - (-1)^{x_i})|x\rangle = x_i|x\rangle$.

QUBO and Ising Hamiltonian for MaxCut

The simplest example is that of Maximum Cut. In this problem the goal is, given a graph $G = (V, E)$, to split the vertex set $V = V_1 \cup (V \setminus V_1)$ in a way that the number of edges (u, v) that are *cut* by the partition (i.e. the edges that have precisely one node is in V_1) is maximal. The QUBO formulation associates one variable per node:

$$\max_{x \in \{0,1\}^n} \sum_{(u,v) \in E} (x_u + x_v - 2x_u x_v) \quad (3.10)$$

and $V_1 = \{i : x_i = 1\}$. The corresponding Ising Hamiltonian is

$$H = - \sum_{(u,v) \in E} \frac{1}{2} (I - Z_u Z_v) \quad (3.11)$$

As expected, QUBO is NP-hard [Pardalos and Jha, 1992]. Nevertheless, in particular cases there exist polynomial-time algorithms. For example, a linear algorithm is presented in [Barahona, 1986] for instances with the connectivity E given by a *series-parallel graph*. We recall that series-parallel graphs are recursively defined by *series* and *parallel compositions* [Eppstein, 1992]. The base case is a graph with a single edge and two distinguished nodes called *source* and *sink*.

QUBO instances with about ≈ 100 variables can be solved exactly by such algorithms as Branch & Bound or Column Generation [Mauri and Lorena, 2012]. Numerous metaheuristics such that tabu search, greedy, genetic, and evolutionary approaches as well as local search methods were adapted to QUBO [Kochenberger *et al.*, 2014]. The most rapid of them can tackle about $\approx 10^4$ variables. It was also observed that *Simulated Annealing* metaheuristic, which is close in spirit to quantum heuristics behaves efficiently compared to other methods [Alkhamis *et al.*, 1998].

3.2.2 Beyond QUBO

QUBO is a very powerful model for optimization problems. However, most natural formulations in combinatorial optimization go beyond QUBO. Indeed, the native formulations of problems of interest may have non-binary integer variables (ex. graph coloring), higher-degree interactions (ex. E3LIN2, Max-Q-XORSAT) or hard constraints that restrict the solution space.

Non-binary bounded integers can be represented by unary encoding [Hadfield *et al.*, 2019] that associated to an integer $x \in [0, \dots, k-1]$ a sequence of bits $x_b \in \{0, 1\}^k$ where the value 1 is placed at the position k . This encoding induces a new hard constraint on the solution space: it is possible to recover the corresponding integer from a bitstring $x_b \in \{0, 1\}^k$ only if it satisfies the condition $\sum_{i=1}^k x_b^i = 1$. Another option is to use qudits - quantum systems that by definition can be in k different states. This approach goes beyond the traditional framework for quantum computing. Despite this, in [Bravyi *et al.*, 2022] *qutrits* were used to solve the Max-3-Cut problem.

In quantum computing, there are two approaches to address *hard constraints*. The first one is *penalization*, i.e. the modification of the QUBO objective function $f(x)$ in a way that makes unfeasible solutions highly suboptimal. For a given constraint that restricts the solution space to S we define a penalty function $g(x) : \{0, 1\}^n \rightarrow \mathbb{R}^+$ such that:

$$g(x) = \begin{cases} 0, & x \in S \\ \geq 1, & x \notin S \end{cases} \quad (3.12)$$

If $f(x)$ is the objective in the initial constrained formulation, the modified objective is $f_S(x) = f(x) + Pg(x)$ where P is a huge number compared to the possible values of f .

Widely-used penalty functions are $g(x) = x_1x_2$ for the constraint $x_1 + x_2 \leq 1$ and $g(x) = (\sum_{i=1}^n a_i x_i - b)^2$ for linear constraints $\sum_{i=1}^n a_i x_i = b$ [Kochenberger *et al.*, 2014]. For instance, with these penalties it is possible to transform an integer linear program for graph coloring in a QUBO [Kochenberger *et al.*, 2005].

Example: The Maximum Independent Set on a graph $G = (V, E)$ is a subset of pairwise disconnected nodes of the biggest cardinality. It is naturally formulated as an integer linear program:

$$\max \sum_{i \in V} x_i \quad (3.13)$$

$$x_i + x_j \leq 1, \quad \forall (i, j) \in E \quad (3.14)$$

$$x_i \in \{0, 1\}, \quad \forall i \in V \quad (3.15)$$

After applying the transformation tricks the problem becomes

$$\max \sum_{i \in V} x_i - P \sum_{(i, j) \in E} x_i x_j \quad (3.16)$$

where it suffices to take $P = n$ [Bomze *et al.*, 1999]. In [Farhi *et al.*, 2020] it was suggested to fix a relatively small $P = 1$ and to extract independent sets from infeasible solutions with a specially-designed postprocessing.

The second technique for constraint management suggests replacing the transverse field $\sum_{v \in V} \sigma_v X_v$ used by quantum algorithms with *another mixing term* that preserves the evolution

in the *feasible subspace* [Hen and Spedalieri, 2016, Hen and Sarandy, 2016, Hadfield *et al.*, 2019]. This approach can significantly reduce the amount of required pair interaction compared to the penalization, which is important for limited-connectivity hardware. Moreover, it is particularly promising for the constraints arising from the unary encoding as the corresponding feasibility condition is preserved by so-called XY-mixers [Hadfield, 2018, Wang *et al.*, 2020]. We remind that XY-Hamiltonian on k qubits is

$$H_{XY} = \sum_{(i,j) \in E} X_i X_j + Y_i Y_j \quad (3.17)$$

where the connectivity E usually is either a chain or a complete graph.

Some hardware-specific constraint embedding strategies were studied into [Vyskocil and Djidjev, 2019] for linear equality constraints. Linear inequalities can be transformed in equalities by introducing slack variables that, in turn, have to be represented with a binary expansion. A hybrid quantum-classical method that doesn't require slack variables was introduced in [Yonaga *et al.*, 2020]. This method solves QUBOs as a part of the iterative procedure.

Finally, natural objective functions for some important problems such as E3LIN2 or Max-K-XORSAT involve *higher-degree interactions*. Using additional variables they can always be reduced to QUBO [Anthony *et al.*, 2017] but such reduction leads to a significant increase in the amount of needed resources. As the number of qubits is strictly limited on NISQ machines, the approaches that deal *directly* with higher-degree functions may be more practical [Glos *et al.*, 2022].

Therefore, in general settings quantum algorithms deal with any pseudo-boolean cost function $C : \{0, 1\}^n \rightarrow \mathbb{R}$ encoded by some diagonal Hamiltonian:

$$\hat{C} : \hat{C}|x\rangle = C(x)|x\rangle \quad (3.18)$$

The work [Hadfield, 2021] provides multiple examples of Hamiltonians encoding boolean and real-valued functions. In the following we will assume that instances of optimization problems are given by Hamiltonian operators C satisfying the condition (3.18).

3.3 Quantum Adiabatic Algorithm

Adiabatic Algorithm for combinatorial optimization was originally introduced in [Farhi *et al.*, 2000]. In the Adiabatic Algorithm the quantum system evolves under the weighted sum of two Hamiltonians H_M and H_C . For a problem with n binary variables these are a *transverse-field* $H_M = \sum_{i=1}^n X_i$ and a *problem Hamiltonian* $H_C : H_C|x\rangle = f(x)|x\rangle$ where $f(x)$ is the function that we want to optimize.

The quantum evolution of total duration t_f is governed by the Hamiltonian $H(t)$:

$$H(t) = (1 - A(t))H_C + A(t)H_M \quad (3.19)$$

The Hamiltonian $H(t)$ with diagonal H_C and H_M being transverse field is *stoquastic*, i.e. its off-diagonal elements are non-positive.

We assume that the time-dependent Hamiltonian may be written in a form:

$$H(t) = \tilde{H} \left(\frac{t}{t_f} \right) = \tilde{H}(s) \quad (3.20)$$

where t_f is the total time of evolution.

A system quantum system initialized in the state $|\psi_0\rangle$ evolves according to the Schroedinger equation:

$$i \frac{\partial}{\partial s} |\psi(s)\rangle = t_f \tilde{H}(s) |\psi(s)\rangle \quad (3.21)$$

$$|\psi(0)\rangle = |\psi_0\rangle \quad (3.22)$$

where $\tilde{H}(s) = (1 - \tilde{A}(s))H_C + \tilde{A}(s)H_M$ (for simplicity we took $\hbar = 1$).

In traditional Adiabatic Algorithm the initial state is taken to be the ground state of H_M : $|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$, and the control function $\tilde{A}(s)$ gradually changes from 1 at $s = 0$ to 0 at $s = 1$. Another version suggests to gradually *turn off the transverse field*:

$$\tilde{H}(s) = H_C + \Gamma(s)H_M \quad (3.23)$$

where $\Gamma(s)$ decreases from a big value G_0 to 0 at $s = 1$. Sometimes the adiabatic evolution under (3.23) is called *quantum annealing* that we distinguish from the heuristic with the same name presented in section 3.4.

Adiabatic theorem

According to the *adiabatic theorem*, if the evolution is slow enough the final state of the system is close to the ground state of $\tilde{H}(1) = H_C$ [Jansen *et al.*, 2007, Amin, 2009a].

There exists several formulation of the adiabatic theorem with different conditions on the Hamiltonian's smoothness. We refer to the survey [Albash and Lidar, 2018] for an comprehensive review of different formulations.

The work [Amin, 2009b] gives a formulation that doesn't impose any specific conditions on $\tilde{H}(s)$. This formulation claims that the system remains in a ground state if the following condition is satisfied:

$$\frac{1}{t_f} \max_{s \in [0,1]} \frac{|\langle \epsilon_i(s) | \partial_s \tilde{H}(s) | \epsilon_j(s) \rangle|}{|\epsilon_i(s) - \epsilon_j(s)|^2} \ll 1, \quad \forall i \neq j \quad (3.24)$$

where $|\epsilon_j(s)\rangle$ is an eigenstate of $\tilde{H}(s)$ with the eigenvalue $\epsilon_j(s)$. The index j corresponds to the relative order of the energy level in the spectrum: $\epsilon_1(s) < \epsilon_2(s) < \dots$. The difference $\Delta(s) = \epsilon_2(s) - \epsilon_1(s)$ is often referred as *spectral gap*. A informal version of the adiabatic theorem tells that the total evolution time t_f should be big with respect to the inverse of the gap square $\frac{1}{\Delta^2}$ in order to preserve the evolution in the instantaneous ground state.

Universality of adiabatic evolution

It was proven that the adiabatic evolution with local non-stoquastic Hamiltonians is universal [Aharonov *et al.*, 2007]. It means in particular that any state reachable by the gate-based quantum computer in time T is the ground state of some local H'_C and the gap $\Delta(s)$ of the transverse-field Hamiltonian $H(s)$ is polynomial at any moment s .

However, as Hamiltonians involved in the Adiabatic Algorithm for optimization are stoquastic it is unlikely that the algorithm allows a universal quantum computation [Albash and Lidar, 2018]. From the complexity point of view, it was shown that for any stoquastic Hamiltonian H_C such that the spectral gap of $\tilde{H}(s)$ is polynomial the ground state problem belongs to PostBPP [Bravyi *et al.*, 2006].

It is an open question if there are some instances H_C such that the gap $\Delta(s)$ is polynomial while classical algorithms take exponential time to find the solution. As quantum computers are not believed to solve NP-complete problems in polynomial time, the Adiabatic Algorithm is usually tested for *limited quantum advantage* [Albash and Lidar, 2018]. It means that the algorithm is not challenged by *the best possible* classical algorithm or even by *the best already known* algorithm but, instead, it is compared to classical metaheuristics inspired by similar ideas.

3.3.1 Classical annealing

The Adiabatic Algorithm is often compared to two classical algorithms: *Simulated Annealing* and *Simulated Quantum Annealing*.

Simulated Annealing

Simulated Annealing is an acknowledged metaheuristic inspired by the physical procedure of the heating and *slow cooling* of a metal to a uniform crystalline state [Kirkpatrick *et al.*, 1983]. In Simulated Annealing the cost function $f(x)$ is identified with the energy of a statistical-mechanical system. The role of the transverse field H_M is played by the kinetic energy that is proportional to the system's temperature $T(t)$. The evolution starts at high temperature T_0 and is slowly cooled to $T(t_f) = 0$. Thermal fluctuations, just as quantum tunneling in the quantum annealing process, help the system to explore the entire energy landscape. In particular, they allow escaping from suboptimal local minima by changing the configuration to one with higher potential energy. As the temperature gets lower, such transitions become less likely and at the end the system is in the lowest-energy state.

Just as the *adiabatic condition* guarantees the convergence to the optimum for the Adiabatic Algorithm, the Simulated Annealing is promised to terminate in a ground state if the system is maintained in the *thermal equilibrium* at every moment. It was shown in [Geman and Geman, 1984] that if the temperature decreases as

$$T(t) \propto \frac{n}{\log(\alpha t + 1)} \quad (3.25)$$

where n is the amount of variables the thermal equilibrium condition is satisfied. A similar result was derived for the adiabatic condition [Morita and Nishimori, 2008]: the condition is satisfied if the transverse field decreases as

$$\Gamma(t) \propto \frac{1}{(\delta t)^{1/(2n-1)}} \quad (3.26)$$

where the parameter δ depends on the already mentioned spectral gap Δ .

The decrease of quantum fluctuations (3.26) obeys a power law which is asymptotically faster than the logarithmic decrease (3.25) for the Simulated Annealing. This inspires a certain optimism in quantum annealing potentially improving over Simulated Annealing [Morita and Nishimori, 2008]. The work [Farhi *et al.*, 2002] demonstrates instances where the Adiabatic Algorithm doesn't provide any improvement over the Simulated Annealing as well as instances showing limited quantum advantage.

Simulated Quantum Annealing

The programmed implementations of simulated annealing imitate the physical cooling process. Likewise, *Simulated Quantum Annealing (SQA)* was designed to emulate the quantum evolution

that obeys the Schroedinger equation (3.22) on a *classical computer* [Martoňák *et al.*, 2002]. The idea is realized by Quantum Monte-Carlo Simulation that modifies a set of configurations imitating the *quantum tunneling* phenomena. A peculiar result [Morita and Nishimori, 2008] claiming for the SQA the same convergence condition (3.26) as for the actual quantum annealing revitalizes the debate of their relative power. A family of instances that are polynomially solved by the Adiabatic Algorithm but not with SQA were presented in [Hastings and Freedman, 2013].

Regarding the relation between SQA and Simulated Annealing, SQA was shown to perform better on random spin-glass [Martoňák *et al.*, 2002], traveling salesman problem [Martoňák *et al.*, 2004]. However, on random instances of 3-SAT Problem the vanilla SQA obtained worse results than the thermal annealing [Battaglia *et al.*, 2005]. However, the modification of the annealing schedule can inverse the situation back in favor of SQA.

Several modifications were suggested that can improve the performance of the Adiabatic Algorithm [Albash and Lidar, 2018, Crosson *et al.*, 2014]. Some of them suggest increasing the spectral gap by the addition of a local field in the middle of the evolution [Crosson *et al.*, 2014] or by modifying the annealing schedule $\tilde{A}(s)$ [Morita and Nishimori, 2008]. Others suggest abandoning adiabaticity either by the initialization in an excited state [Crosson *et al.*, 2014] or by allowing *diabatic transitions*.

3.4 Quantum Annealing

Quantum Annealing was applied to a lot of optimization problem such as Traveling Salesman and graph coloring [Rieffel *et al.*, 2014], image recognition [Neven *et al.*, 2008] and the ground state search for the Sherrington-Kirkpatrick model [Venturelli *et al.*, 2015]. In essence, the idea behind the Quantum Annealing is to use *quantum fluctuations* in order to explore the solution space.

In Quantum Annealing the system's state evolves according to the Schroedinger equation (3.22) under the Hamiltonian $H(t) = a(t)H_C + b(t)H_M$ where, just as in the Adiabatic Algorithm, H_C encodes the cost function and H_M is a transverse field $H_M = \sum_{i=1}^n X_i$. There are two traditional choices for the schedule coefficients $a(t)$ and $b(t)$. The first option is to chose the coefficients as in the Adiabatic Algorithm ($a(t) = \frac{t}{t_f}$, $b(t) = 1 - \frac{t}{t_f}$). The other approach consists in fixing a constant $a(s) = C$ and to gradually "turn off" the transverse field : from large $b(0) \gg 0$ to $b(t_f) = 0$.

Analog quantum hardware

The annealing process requires a special analogue device called *quantum annealer* that allows continuous control of the Hamiltonian of the system. The most famous such devices are manufactured by DWave on superconducting qubits, while there are also realizations on Rydberg's atoms [Pichler *et al.*, 2018]. Such physical systems allow only pair interactions so Quantum Annealing is used to solve problems only encoded by an Ising model.

Due to the decoherence effect of the physical hardware, a hard time limit is imposed on the annealing time $t_f < \tau$ where τ is the coherence time. The limited runtime implies that the adiabatic condition (3.24) is generally not satisfied. Therefore, the system can jump in an excited state and return a suboptimal result in the end. *Quantum Annealing* is an *heuristic*, just as widely used implementations of Simulated Annealing with fixed cooling times.

The potential profit comes from the quantum tunneling that can, in principle, help the system to escape from local minima (see figure 3.2). In *Simulated Quantum Annealing* that also exploits

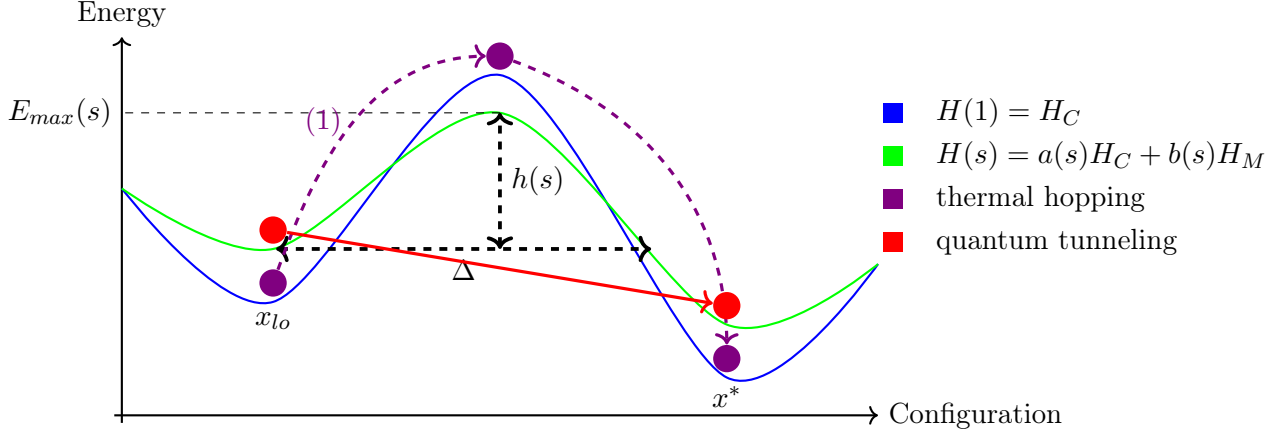


Figure 3.2: Quantum tunneling versus thermal hopping through a potential barrier.

In *Simulated Annealing* the potential energy landscape $E = H_C$ (blue curve) corresponds to the cost function. A system trapped in a local minima x_{lo} can escape by thermal hopping (violet path): first go to a higher-energy state and then drop in a global minima x^* . The probability of the transition (1) to a higher-energy state decreases with the temperature T .

In *Quantum Annealing* the energy at some moment s (green curve) is an interpolation $H(s) = a(s)H_C + b(s)H_M$ of the problem Hamiltonian H_C and a transverse field H_M . The coefficient $b(s)$ decreases to zero at the final moment $s = 1$ and the final energy operator encodes exactly the cost function $H(1) = H_C$ (blue curve). In quantum mechanics a system with energy $E < E_{max}(s)$ can directly tunnel through the energy barrier (red path). The probability of the tunneling depends on the width Δ and the height $h(s)$ of the barrier. In Quantum Annealing the height $h(s)$ is gradually increased by tuning $a(s)$ and $b(s)$, so the tunneling probability gets lower.

Simulated Quantum Annealing imitates the quantum tunneling inside a classical algorithm.

the tunneling effect, the quantum evolution under the Schroedinger equation is stochastically approximated by a Quantum Monte-Carlo chain. It was shown that quantum effects lead to a more efficient exploration of the search space than the thermal hopping [Kadowaki and Nishimori, 1998, Martoňák *et al.*, 2002, Martoňák *et al.*, 2004].

3.4.1 Performance scaling for different annealing times

An interesting question is how the algorithm behaves when the runtime limit t_f changes. For practical applications the success probability is a particularly relevant metric. The success probability is defined by the overlap $|\langle x^* | \psi_{t_f} \rangle|^2$ of the final state $|\psi(t_f)\rangle$ with the ground state $|x^*\rangle$. According to the adiabatic theorem at the infinite time limit $t_f \rightarrow \infty$ the overlap tends to 1, but somewhat surprisingly it doesn't grow monotonically. A numerical simulation reported in [Crosson *et al.*, 2014] reveals that, given a range $t_f \in [T_{min}, T_{max}]$, there is an optimal T^* that depends on the considered instance such that the success probability on $[T_{min}, T_{max}]$ is maximal in T^* . For most instances of the Max-2-Sat problem considered in [Crosson *et al.*, 2014] the optimal T^* is *strictly inside* the interval. Hence, the solution doesn't necessarily get better with

a finite increase of t_f .

Lieb-Robinson bound

At the moment, there are only a few works with analytical results about Quantum Annealing in the constant time regime $t_f = O(1)$. The reason is a lack of tools highlighted in [Braidia *et al.*, 2022]. Generally, theoretical analyses are based on the *Lieb-Robinson bound*. This bound allows approximating the impact of the whole system on a local quantum observable by the action of its local neighborhood [Tran *et al.*, 2019]. More precisely, the bound applies to a local Hamiltonian $H = \sum_{i=1}^k H_k$ where each term acts on a small subset of qubits S_k . The Hamiltonian's connectivity is given by a graph $G = (V, E)$ where each node corresponds to a qubit and there is an edge between u and v if there is a term H_k that acts on both $u \in S_k, v \in S_k$. We call the r -neighborhood of a term H_k all qubits that are at distance at most r from S_k in the graph G . The Lieb-Robinson bound implies that for a short time t_f we can approximate:

$$\langle \psi(t_f) | H_k | \psi(t_f) \rangle \approx \langle \psi_r(t_f) | H_k | \psi_r(t_f) \rangle \quad (3.27)$$

where $|\psi_r(t)\rangle$ is a restriction of the system state $|\psi(t)\rangle$ on the r -neighborhood of H_k . The size r of the relevant neighborhood depends on the time t_f . This dependence is sometimes called the *light cone* and is interpreted as a speed of information transmission in the quantum evolution. The bound can be applied for sparse connectivity G and short runtime t_f when the local dynamic can be simulated classically in a reasonable time.

In [Braidia *et al.*, 2022] the Lieb-Robinson bound was used to establish an approximation ratio guarantee for the Quantum Annealing with very short t_f on MaxCut on triangle-free 3-regular graphs. However, the obtained guarantee is not expected to be tight. Besides, the work [Moosavian *et al.*, 2022] uses the bound to expose a set of instances for which the Quantum Annealing is not believed to show a quantum advantage in any constant time.

3.4.2 Experimental performance

In opposition to a limited number of theoretical results, a lot of efforts are invested in the *experimental evaluation* of Quantum Annealing. It is mostly due to the availability of large DWave machines that since recently can be accessed on a cloud. the question if they really implement quantum effects is still a subject of active debate with a number of negative [Shin *et al.*, 2014, Martin-Mayor and Hen, 2015, Rønnow *et al.*, 2014] and positive pieces of evidence [Boixo *et al.*, 2014]. Interestingly, the study [Boixo *et al.*, 2014] claims the presence of quantum effects by showing that DWave's output strongly correlates with those of SQA while having different results with the thermal annealing simulation.

Quantum annealers suffer from multiple constraints, some of them due to the construction choices and others - to unavoidable physical phenomena.

Embedding

One of the potential obstructions for the applicability of DWave annealer is its *sparse connectivity*: a physical qubit in the chip interacts with its neighbors in a *Chimera* or *Pegasus* sparse layout [Systems,]. Therefore, the Ising model for the cost function has to be *embedded* in the hardware architecture. The embedding associates to a *logical qubit* a chain of *physical qubits* that are encouraged to take the same value by ferromagnetic coupling $J_{i,j} < 0$.

For any initial connectivity pattern the embedding can be done with at most quadratic overhead in the number of qubits [Choi, 2008]. An example for a fully-connected random Ising model is given in [Venturelli *et al.*, 2015]. On the downside, the embedding reduces the performance of the heuristic by adding *an extra energy scale* required to impose the chain coupling $J_{i,j} < 0$. It also significantly limits the size of the instances that can be treated by the machine as $O(n^2)$ qubits are needed to embed a fully connected model on n variables. Hence it is somewhat more important to increase the hardware connectivity than to increase the number of qubits [Rieffel *et al.*, 2014].

In instances issued from combinatorial optimization, it is often the case that dense connectivity is a result of penalization of hard constraints. Therefore, other techniques to deal with constraints are of great interest. The approach from [Hen and Spedalieri, 2016, Hen and Sarandy, 2016] suggests modifying the *mixing term* in order to keep the evolution in the feasible subspace. Such a modification consists in the initialization of the system in some feasible state and the evolution under new driver H_M . A way to impose a linear equality constraint on n variables using $O(n)$ physical qubits from DWave graph is presented in [Vyskocil and Djidjev, 2019]. For some applications it may be profitable to interchange the embedding order: the work [Bian *et al.*, 2016] on *Constraint Satisfaction Problem* suggests to locally embed the problem's constraints and then couple qubits that correspond to the same variable.

Another obstacle for a good performance is *the limited range and precision of control parameters* $J_{i,j}$ and h_i in the physical Ising model $H = \sum_{(u,v) \in E_H} J_{u,v} Z_u Z_v + \sum_{u \in V_H} h_u Z_u$ where $G_H = (V_H, E_H)$ is the hardware layout graph [Venturelli *et al.*, 2015]. A way to intelligently deal with these restrictions is described in [Bian *et al.*, 2014].

The sparse connectivity and the limited control are in a sense due to the construction choice (that is usually justified by experimental constraints). These characteristics define a set of hardware-native instances that can directly be solved by the annealer. Complications arise when the instance of interest doesn't naturally map to these parameters and needs to be embedded.

Hardware noise

In addition to construction limits, such physical phenomena as *decoherence* and *noise* impact the annealing performance on *all instances* including hardware-native ones. Indeed, it was claimed that a non-zero environment temperature may transform a quantum annealer in a simple thermal annealer [Albash *et al.*, 2017] or make it perform even worse [Martin-Mayor and Hen, 2015].

Besides, the annealer's performance also suffers from *random fields* and *random-bond* fluctuations that are present in the chip. It was shown that in order to assure close-to-correct dynamics while growing the number of qubits a significant reduction of such effects is required [Zhu *et al.*, 2016]. Another option to tackle this issue is to use *error mitigation* [Kandala *et al.*, 2019, Berg *et al.*, 2022].

3.5 Quantum Approximate Optimization Algorithm (QAOA)

QAOA, originally introduced in [Farhi *et al.*, 2014a] is a quantum-classical heuristic for combinatorial optimization. In a same manner that Quantum Annealing, QAOA receives in input a pseudo-boolean cost function $f : \{0,1\}^n \rightarrow \mathbb{R}$ encoded in a diagonal energy operator $C : C|x\rangle = f(x)|x\rangle$. It returns a low-energy state x_o that corresponds to an approximate optimum of $f(x)$.

3.5.1 QAOA components

QAOA uses quantum computers to prepare a state made out of three principal components: an *initial state* $|\psi_0\rangle$ and two families of operators that don't commute called *phase-separation* and *mixing*.

Initial state

The *initial state* $|\psi_0\rangle$ should be in the first place *trivial to prepare*. It is usually taken to be a uniform superposition:

$$|\psi_0\rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} |x\rangle \quad (3.28)$$

Such choice of the initial state maintains the link between QAOA and the Adiabatic Algorithm. However, other options may be beneficial in distinct optimization contexts. For instance, if the problem of interest has hard constraints, a version of QAOA [Hadfield *et al.*, 2019] suggests to explicitly keep the evolution in a feasible subspace $S_b \subsetneq \{0,1\}^n$. In this approach, the system is initialized in some superposition over feasible solutions.

Another alternative is to *warm start* the optimization with a proper choice of the initial state. The idea behind this is to integrate the information used by classical approximation algorithms in order to concentrate the search efforts on promising solutions. For the MaxCut example, the classical information can be the approximate solution returned by the Goemans Williamson optimization algorithm [Egger *et al.*, 2021] or the information gained from the corresponding semidefinite program [Egger *et al.*, 2021, Tate *et al.*, 2020].

Phase separation operator

Phase-separation operators carry the information about the objective function. The family contains the following parametrized diagonal unitaries:

$$U_P(\gamma) = e^{i\gamma C} \quad (3.29)$$

where C is the problem Hamiltonian and γ is the parameter that serves as the rotation angle.

Mixing operators

As phase-operators are diagonal, they alone can't change the probability distribution. Therefore, a *mixing* family is introduced. This family is responsible for the exploration of the solution space. In a traditional QAOA the mixing is defined as:

$$U_M(\beta) = e^{i\beta B} \quad (3.30)$$

where $B = \sum_{i=1}^n X_i$ is the transverse field Hamiltonian.

If the solution space is restricted to $S_b \subsetneq \{0,1\}^n$ by hard constraints other mixing families can be appealing [Hadfield *et al.*, 2019]. Crucially, a mixing family has to *connect* all feasible solutions: for each pair $(x, y) \in S_b$ there should exist $m \in \mathbb{N}$ such that the transition probability is positive:

$$|\langle y | U_M(\beta_{m-1}) \dots U_M(\beta_0) | x \rangle|^2 > 0, \quad \forall x, y \in S_b \quad (3.31)$$

In the Adiabatic Algorithm the evolution starts in the ground state $|\psi_0\rangle$ of the driver Hamiltonian B . Such initialization is a part of the conditions of the adiabatic theorem. In an attempt to preserve the correspondence between QAOA and the Adiabatic Algorithm the mixing family is modified in some warm-start approach [Egger *et al.*, 2021]. A matrix B^{ws} is chosen such that the warm-start state $|\psi_0^{ws}\rangle$ is the ground state of B^{ws} and the mixing family is $U_M(\beta) = e^{i\beta B^{ws}}$.

QAOA applies phase separation and mixing in alternation:

$$|\psi_p(\beta, \gamma)\rangle = U_M(\beta_{m-1})U_P(\gamma_{m-1}) \dots U_M(\beta_0)U_P(\gamma_0)|\psi_0\rangle \quad (3.32)$$

where the number of layers p called *depth* is chosen in advance. For $p \rightarrow \infty$ the QAOA circuit is simply the Trotterization of the adiabatic evolution [Farhi *et al.*, 2014a]. Interestingly, the form (3.32) is *universal*, i.e. it is expressive enough to encode any desired quantum computation [Lloyd, 2018].

At the end the system is measured in the computational basis. The solution $|x\rangle$ is returned with probability $|\langle x|\psi_p(\beta, \gamma)\rangle|^2$.

Classical loop

The classical part of QAOA serves to determine parameter values β^*, γ^* . Each choice of parameters correspond to a different states $|\psi_p(\beta, \gamma)\rangle$ and, in consequence, to a different probability distributions

$$\mathbb{P}_{\beta, \gamma}(x) = |\langle x|\psi_p(\beta, \gamma)\rangle|^2, \quad x \in \{0, 1\}^n \quad (3.33)$$

QAOA output is a sample from the parameterized distribution (3.33). We *score* a set of parameter values β, γ by the ability of the corresponding distribution $\mathbb{P}_{\beta, \gamma}(x)$ to return *low-energy states*.

At first sight, one may suggest to discriminate distributions by the probability $\mathbb{P}_{\beta, \gamma}(x^*)$ to output the optimal solution x^* . This, however, doesn't make sense as such metric requires to *know* the solution that we are looking for in advance. Instead, QAOA compares the distributions by the corresponding expectations of the objective function:

$$L(\beta, \gamma) = \langle f \rangle_{\beta, \gamma} = \langle \psi_p(\beta, \gamma) | C | \psi_p(\beta, \gamma) \rangle \quad (3.34)$$

How many measurements we need?

We expect the algorithm to return a value that is at least as good as the expectation $L(\beta, \gamma)$. The number of repetitions m required to obtain a sample \hat{X} such that $f(\hat{X}) < L(\beta, \gamma) + 1$ can be estimated from the Chebyshev's inequality.

We consider the average energy $\bar{f}_m = \frac{1}{m} \sum_{i=1}^m f(X_i)$ of m independent samples X_1, \dots, X_m from $\mathbb{P}_{\beta, \gamma}$. By linearity,

$$E[\bar{f}_m] = \frac{1}{m} \sum_{i=1}^m E[f(X_i)] = L(\beta, \gamma) \quad (3.35)$$

The Chebyshev's inequality states:

$$\mathbb{P}_{\beta, \gamma} [|\bar{f}_m - L(\beta, \gamma)| \geq 1] \leq \bar{\sigma}^2 \quad (3.36)$$

where $\bar{\sigma}^2$ is the variance of \bar{f}_m . As all X_i are independent $\bar{\sigma}^2 = \frac{\sigma^2}{m}$ where σ^2 is the variance of the mean energy in the QAOA state: $\sigma^2 = \langle f^2 \rangle_{\beta, \gamma} - \langle f \rangle_{\beta, \gamma}^2$.

After m repetitions with probability at least $1 - \frac{\sigma^2}{m}$ the average $\bar{f}_m < L(\beta, \gamma) + 1$. If this inequality is true, there is at least one sample X_k such that $f(X_k) \leq L(\beta, \gamma) + 1$. This sample is the solution returned by QAOA.

3.5.2 How to compute the mean energy?

Accordingly to the previous observations, the mean energy $L(\beta, \gamma)$ is indeed a reasonable choice for the loss function in the classical parameter optimization routine. Yet it is not clear how to *compute* the value $L(\beta, \gamma)$.

The work [Bravyi *et al.*, 2021] speculates about the intrinsic difficulty to evaluate the expectation of a general observable $O = O_1 \otimes \dots \otimes O_n$ in the output distribution of a shallow quantum circuit. For instance, for a constant-depth circuit even the approximate value is #P-hard to compute. In a special case where each O_i is close to identity the problem admits a polynomial approximation scheme. This setting *doesn't* apply to QAOA where the energy operator C is a sum of operators in the form $C_k = \prod_{i \in S_k} Z_i$ [Bravyi *et al.*, 2021].

On the other side, the QAOA circuit has a specific form that can simplify the computation of mean values. Indeed, for the MaxCut problem a formula for the mean value in QAOA with $p = 1$ was presented in [Wang *et al.*, 2017]. We generalized the result to the case of weighted MaxCut in [Dalyac *et al.*, 2021] and, using the ZX-calculus, to the general Ising model (see chapter 8).

Claim 3.5.1 *For the weighted MaxCut problem encoded in the Ising model $C = \sum_{1 \leq u < v \leq n} e_{u,v} Z_u Z_v$ the mean value of $Z_u Z_v$ in the parameterized state $|\psi(\beta, \gamma)\rangle = U_M(\beta) U_C(\gamma) |\psi_0\rangle$ of QAOA with depth 1 is:*

$$\begin{aligned} \langle \psi(\beta, \gamma) | Z_u Z_v | \psi(\beta, \gamma) \rangle &= \frac{-\sin(4\beta) \sin(\gamma e_{u,v})}{2} \left[\prod_{w \neq u,v} \cos(\gamma e_{u,w}) + \prod_{w \neq u,v} \cos(\gamma e_{v,w}) \right] + \\ &+ \frac{\sin^2(2\beta)}{2} \left[\prod_{w \neq u,v} \cos(\gamma(e_{u,w} - e_{v,w})) - \prod_{w \neq u,v} \cos(\gamma(e_{u,w} + e_{v,w})) \right] \end{aligned} \quad (3.37)$$

We remark that the same result was obtained in an independent work [Bravyi *et al.*, 2020].

The works [Basso *et al.*, 2021, Farhi *et al.*, 2022] suggest iterative procedures that compute mean energies. The former considers MaxCut problem on *large-girth* (no short cycles) regular graphs and the latter investigate the Sherrington-Kirkpatrick model in infinite size limit. Both procedures have the complexity that grows exponentially with p , although the complexity scalings are independent from the number of variables n . In [Hadfield *et al.*, 2021] the mean values for different p are written as power series.

Stochastic approximation

In all other cases one can access the expectation value $L(\beta, \gamma)$ using stochastic approximation:

$$L(\beta, \gamma) \approx \hat{L}_N^{\beta, \gamma} = \frac{1}{N} \sum_{k=1}^N f(x_k) \quad (3.38)$$

If the stochastic approximation is used, the classical routine that minimizes $L(\beta, \gamma)$ over possible values of β and γ makes calls to the quantum computer to get samples X_1, \dots, X_N . Other

stochastic aggregation function such as *Conditional Value-at-Risk* can be employed to score parameter values [Braine *et al.*, 2021] with a potential improvement of the convergence speed [Barkoutsos *et al.*, 2020].

The interaction between quantum and classical parts of the algorithm is demonstrated on the figure 3.3. To resume, QAOA use quantum computer in two separate contexts. The classical procedure that search optimal parameters makes calls to the quantum processor when the loss function $L(\beta, \gamma)$ is evaluated stochastically. Once optimal parameters β^*, γ^* are determined, the QAOA state is measured several times and the best output is returned as solution.

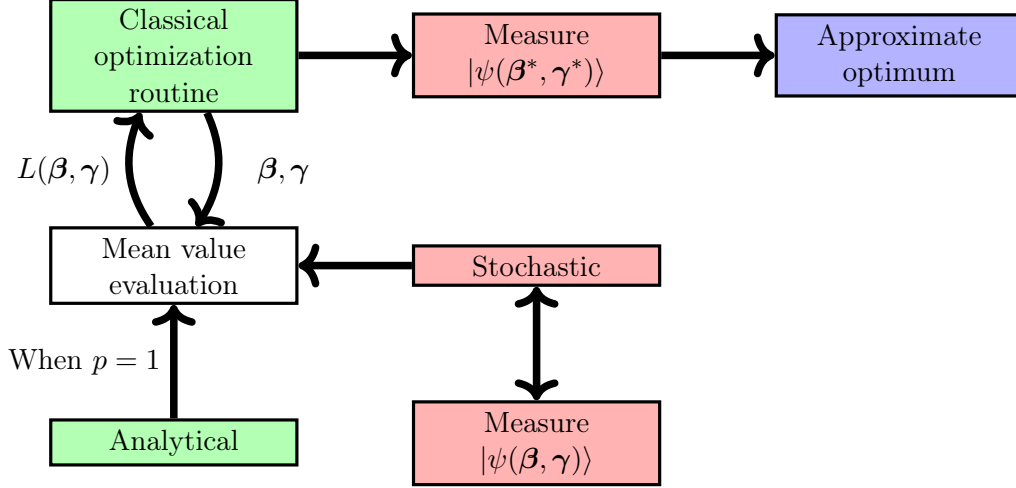


Figure 3.3: The interaction between classical and quantum parts of QAOA. Green (red) rectangles correspond to classical (quantum) parts of the algorithm. Mean value evaluation depends on the implementation and may either make calls to a quantum computer or, when $p = 1$ and in some special cases be done classically.

The optimal choice of parameter lead to the following guarantee on the approximation ratio of QAOA:

$$\alpha = \frac{\min_{\beta, \gamma} L(\beta, \gamma)}{f(x^*)} \quad (3.39)$$

where x^* is the optimal solution of the problem. In practice, the parameter optimization search is difficult because of the highly irregular landscape with *multiple local minimas*. If the loss function $O(\beta, \gamma)$ is evaluated stochastically, the optimization is hampered by the related stochastic noise.

For the reasons mentioned above the optimization procedure returns values β^*, γ^* such that $L(\beta^*, \gamma^*) \geq \min_{\beta, \gamma} L(\beta, \gamma)$. The contribution of the optimization error can be explicitly separated in the performance metric:

$$\epsilon = \frac{L(\beta^*, \gamma^*) - f(x^*)}{f_{max} - f(x^*)} \quad (3.40)$$

$$= \underbrace{\frac{L(\beta^*, \gamma^*) - \min_{\beta, \gamma} L(\beta, \gamma)}{f_{max} - f(x^*)}}_{E_M} + \underbrace{\frac{\min_{\beta, \gamma} L(\beta, \gamma) - f(x^*)}{f_{max} - f(x^*)}}_{E_O} \quad (3.41)$$

In the expression (3.41) the errors E_M and E_O are due to the *model mismatch* and to the *optimization error* respectively.

3.5.3 Performance guarantees for QAOA

For the big number of layers p QAOA circuits are able to reproduce the Trotterized adiabatic evolution. Hence, in the limit $p \rightarrow \infty$ the algorithm achieves the exact optimum. Moreover, contrary to Quantum Annealing, the increase in quality of the solution is *monotone*. This fact is a direct consequence of $U_C(0) = U_M(0) = I$:

$$\begin{aligned} \min \langle \psi(\beta_1, \dots, \beta_{p-1}) | C | \psi(\beta_0, \dots, \beta_{p-1}) \rangle &= \min \langle \psi(\beta_1, \dots, \beta_{p-1}, \mathbf{0}) | C | \psi(\beta_1, \dots, \beta_{p-1}, \mathbf{0}) \rangle \\ &\geq \min \langle \psi(\beta_1, \dots, \beta_p) | C | \psi(\beta_1, \dots, \beta_p) \rangle \end{aligned} \quad (3.42)$$

At low depths p the analytical analysis allowed to establish some results about the approximation ratio of QAOA. The first result that caused a lot of excitement was the approximation ratio of $\left(\frac{1}{2} + \frac{1}{22D^{3/4}}\right)$ achieved by QAOA with depth $p = 1$ on E3LIN2 [Farhi *et al.*, 2014b]. At the moment the best known classical guarantee was $\left(\frac{1}{2} + \frac{C}{D}\right)$ so QAOA₁ demonstrated quantum speedup. Since then a better classical algorithm was found with $\alpha = \frac{1}{2} + \frac{C}{D^{1/2}}$ [Barak *et al.*, 2015]. This ratio matches the asymptotic scaling of the inapproximability bound [Trevisan, 2001].

MaxCut

Most of the analytical results were derived for QAOA applied to MaxCut. In MaxCut one aims to split the vertices of a graph $G = (V, E)$ on two sets $V = V_1 \cup V_2$ such that the number of crossing edges $|\{(u, v) \in E : u \in V_1, v \in V_2\}|$ is maximized.

The computation of the approximation ratio with formulas (3.39) and (3.41) requires knowing the exact optimum that is generally difficult to compute. Hence the performance is usually reported in terms of the *cut fraction* c , i.e. the ratio of the size of QAOA cut to the total number of edges $c = \frac{L(\beta^*, \gamma^*)}{|E|}$. The original paper [Farhi *et al.*, 2014a] demonstrated that QAOA₁ achieves the cut fraction of at least 0.6924 on 3-regular graph. On triangle-free D -regular graph QAOA₁ obtains $c \geq \frac{1}{2} + \frac{0.3032}{\sqrt{D}}$ [Hadfield, 2018]. The work [Wurtz and Love, 2021] provides the cut fractions for $p = 2$ and $p = 3$ for 3-regular graphs with no cycles of length 5 and 7 respectively.

Most remarkably, the iterative procedure from [Basso *et al.*, 2022] permitted to show that QAOA for $p \geq 11$ outperforms the *Goemans-Williamson* algorithm on D -regular graphs with girth $l > 2p+1$. The *Goemans-Williamson* algorithm is particularly important as its performance matches the inapproximability bound for *general* graphs (that are not necessarily large-girth or regular). We remark that for the *specific* instances of large-girth regular graphs a polynomial approximation scheme exists [Alaoui *et al.*, 2021], albeit its guarantee relies on a still unproven conjecture.

The above results hold for the *ideal* quantum computer. When a real hardware is used, additional obstructions may significantly degrade the performance.

3.5.4 QAOA circuit

In physical implementation the evolution (3.32) has to be compiled into a circuit composed out of elementary gates. The set of elementary gates contains one-qubit and two-qubit gates that are supported by the hardware. Due to the limited connectivity, only a restricted subset of qubits can directly interact - which requires the addition of SWAP gates for two-qubit interactions that are not connected in the native layout.

We illustrate the compilation on a simple optimization problem encoded in the Hamiltonian:

$$C = \sum_{(u,v) \in E} Z_u Z_v \quad (3.43)$$

where the interaction set E corresponds to the graph $G = (V, E)$ shown on Figure 3.4. Up to a irrelevant additive constant the Hamiltonian (3.43) encodes the MaxCut problem on the graph G .

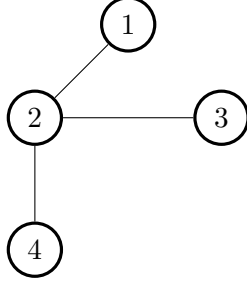


Figure 3.4: Example of the optimization problem.

As the Pauli-Z operators are diagonal the summands of (3.43) mutually commute, just as X_i operators acting on different qubits. Hence we can write:

$$U_C(\gamma) = e^{i\gamma \sum_{(u,v) \in E} Z_u Z_v} = \prod_{(u,v) \in E} e^{i\gamma Z_u Z_v} \quad (3.44)$$

$$U_M(\beta) = e^{i\beta \sum_{i=1}^n X_i} = \prod_{i=1}^n e^{i\beta X_i} \quad (3.45)$$

If we assume that elementary gate set includes the Hadamard gate H , CNOT and the Z -rotation $R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$ for arbitrary angle $\gamma \in [0, 2\pi]$ we can compile $e^{i\beta X_u}$ to the circuit:

$$q_u : \text{---} [H] \text{---} [R_z(2\gamma)] \text{---} [H] \text{---} \quad (3.46)$$

and $e^{i\gamma Z_u Z_v}$ to

$$\begin{array}{c} q_u : \text{---} \bullet \text{---} \text{---} \bullet \text{---} \\ q_v : \text{---} \oplus \text{---} [R_z(2\gamma)] \text{---} \oplus \text{---} \end{array} \quad (3.47)$$

Finally, the circuit for one QAOA layer $U_M(\beta)U_C(\gamma)$ for the Hamiltonian (3.43) is:

$$(3.48)$$

An optimized compilation strategy for the QAOA circuit on hardware with limited connectivity is presented in [Lotshaw *et al.*, 2022, Crooks, 2018].

In general, vanilla QAOA admits a relatively intuitive circuit representation. However, modifications of the mixing operator that are used to impose the hard constraints substantially increase the difficulty of the task. Indeed, even the simplest XY version requires the compilation of $U_M(\beta) = e^{i\beta B}$ where

$$B = \sum_{i=1}^k X_i X_{i+1} + Y_i Y_{i+1} \quad (3.49)$$

Terms in the sum don't commute, so there is no simple way to decompose $U_M(\beta)$ on elementary gates. Although some compilation strategies were proposed [Hadfield, 2018], this obstacle still seriously constrains the capacity of both numerical and physical experimentations.

3.5.5 Hardware implementations

QAOA was implemented on *Google Sycamore* superconducting hardware [Harrigan *et al.*, 2021] for instances with up to 23 variables. Experiments using different platforms were performed on 2-qubit photonic computer [Qiang *et al.*, 2018] and on 40 trapped-ion qubits [Pagano *et al.*, 2020].

It was observed that limited connectivity together with noise dramatically impacts the performance of QAOA [Harrigan *et al.*, 2021]. The question if the quantum advantage is possible at all in the presence of noise is still a matter of debate.

On the fundamental side, the work [França and García-Patrón, 2021] shows that for the current noise level the output distribution is close to a thermal state with $T > T_0$. As a thermal state for $T > T_0$ (T_0 depends on the instance) can be efficiently simulated on a classical computer the noise level should be significantly reduced in order to leave room for any quantum advantage.

The work [Guerreschi and Matsuura, 2019] reports an attempt of a QAOA versus *classical exact solvers* comparative study that takes into account as much realistic conditions as possible. It was observed that, even if asymptotically QAOA (with approximate solution) is faster than exponential *exact* algorithms, it may require hundreds or thousands of qubits in order to get a *runtime improvement*. The stochastic noise in the evaluation of the loss function also negatively affects the overall runtime scaling [Lotshaw *et al.*, 2022].

The situation is slightly more optimistic for hardware-native instances i.e. instances whose connectivity pattern reproduces the qubit layout of the platform. Both works [Harrigan *et al.*, 2021, Pagano *et al.*, 2020] observed that for these instances performance remains constant with the number of variables n while it decreases for instances that require embedding. This is consistent with the intuition: embedding strategies increase the depth of the circuit and longer circuits are more impacted by noise.

Classical simulation of QAOA

Even if some results on the real quantum hardware start to appear, most of the numerical results are obtained using simulations. As the full-scale simulation of the circuit on n qubits has the complexity 2^{2n} the experiments were done for only limited problem sizes.

From the fundamental side, even at the lowest depth $p = 1$ QAOA exhibits *quantum supremacy* [Farhi and Harrow, 2016]. More precisely, it was demonstrated that if one could efficiently sample from the distribution $\mathbb{P}_{\beta_1, \gamma_1}(x)$ the polynomial hierarchy would collapse. As this is unlikely, it is believed that the output of QAOA can't be efficiently simulated. However, in some specific conditions of *small angles* an efficient simulation is possible [Hadfield *et al.*, 2021].

3.5.6 Locality in QAOA

One of the principal features of bounded depth QAOA is its *locality*. Roughly speaking, in a local optimization algorithm whenever it is quantum or classical a variable can be explicitly impacted only by its neighbors [Hastings, 2019]. We recall that the similar property for the Quantum Annealing follows from the Lieb-Robinson bound.

More precisely, a Hamiltonian $H = \sum_{i=1}^N H_i$ is called k -local if each term H_i acts non-trivially on at most k qubits S_k ($|S_k| \leq k$) [Albash and Lidar, 2018]. Two variables u and v are called neighbors if they appear together in at least one S_k . The Ising cost operator

$$C = \sum_{u \in V} h_u Z_u + \sum_{(u,v) \in E} J_{u,v} Z_u Z_v \quad (3.50)$$

is 2-local and the neighborhood relationship is defined by the edges of the connectivity graph $G = (V, E)$. The corresponding phase separation is a product of $e^{i\gamma h_u Z_u}$ and $e^{i\gamma J_{u,v} Z_u Z_v}$ and the only 2-qubit gates that appear in the QAOA circuit act on neighboring pairs of qubits.

This fact has some curious consequences.

Let the notation $\langle O \rangle_p$ stand for the expectation of an observable O in some QAOA state of depth p for arbitrary fixed $\beta, \gamma \in \mathbb{R}^p$. Locality implies that the mean values $\langle Z_u \rangle_p$ and $\langle Z_u Z_v \rangle_p$ depend only on the qubits that are at distance at most p from the nodes u or v in the graph $G = (V, E)$. Indeed, in the Heisenberg representation the expectation of an observable O in a state $|\psi\rangle = U|\psi_0\rangle$ is semantically equivalent to the mean value of the conjugated observable $\hat{O} = U^\dagger O U$ in the state $|\psi_0\rangle$:

$$\langle \psi | O | \psi \rangle = \langle \psi_0 | U^\dagger O U | \psi_0 \rangle = \langle \psi_0 | \hat{O} | \psi_0 \rangle \quad (3.51)$$

Support of an observable

We call *support* of an operator O a set of qubits on which O acts non-trivially. The expectation of the observable O with support Q in a tensor product state $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$ is independent on states $|\psi_i\rangle, i \notin Q$ that are not in the support:

$$\begin{aligned} \langle \psi_n | \otimes \cdots \otimes \langle \psi_1 | O | \psi_1 \rangle \otimes \cdots \otimes |\psi_n\rangle &= \left[\prod_{i \notin Q} \langle \psi_i | \psi_i \rangle \right] \times (\otimes_{i \in Q} \langle \psi_i |) O (\otimes_{i \in Q} |\psi_i\rangle) \\ &= (\otimes_{i \in Q} \langle \psi_i |) O (\otimes_{i \in Q} |\psi_i\rangle) \end{aligned} \quad (3.52)$$

An observable O with support $Q \subset V$ commutes with $e^{i\beta X_u}$ for all $u \notin Q$ and with $e^{i\gamma Z_u Z_v}$ for $u, v \notin Q$. Therefore, the mixing operator doesn't change the support of an observable. When one iteratively conjugates the observable O with QAOA layers:

$$O^i = U_C^\dagger(\gamma_i) U_M^\dagger(\beta_i) O^{i-1} U_M(\beta_i) U_C(\gamma_i) \quad (3.53)$$

the support is extended from Q^{i-1} to $Q^i = Q^{i-1} \cup \{v \in V | \exists u \in Q^{i-1} : (u, v) \in E\}$. After p conjugations the support Q^p for $Z_u Z_v$ contains only qubits at distance at most p from u or v . With a standard initial state $|\psi_0\rangle = |+\rangle \otimes \cdots \otimes |+\rangle$ the expectation is:

$$\langle Z_u Z_v \rangle_p = (\otimes_{i \in Q^p} \langle + |) U_{Q^p}^\dagger(\beta, \gamma) Z_u Z_v U_{Q^p}(\beta, \gamma) (\otimes_{i \in Q^p} |+\rangle) \quad (3.54)$$

where $U_{Q^p}(\beta, \gamma)$ is the restriction of QAOA on the qubits in Q^p (i.e. gates that act *only* on qubits from Q^p).

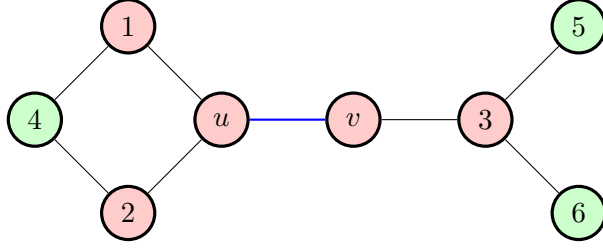


Figure 3.5: The support of the observable $Z_u Z_v$ (blue link) for $p = 1$ (red nodes) and $p = 2$ (red and green nodes).

The role of locality in QAOA

Locality explains the loss function *concentration* phenomenon reported in [Brandao *et al.*, 2018]. The concentration means that for different instances of MaxCut generated according to the same distribution (for example D -regular graphs of different sizes n_1, \dots, n_M) the loss function values $L_1(\beta, \gamma) \dots L_M(\beta, \gamma)$ for fixed β, γ are approximately the same across instances.

Most of the results limiting the performance of QAOA also follows from the locality. The work [Hastings, 2019] shows the cases when *local classical* algorithms outperform single-layer QAOA. It also conjectures that a bounded-depth QAOA can't improve over global classical methods on E3LIN2. Works [Farhi *et al.*, 2020] and [Bravyi *et al.*, 2020] use locality to bound the approximation ratio of constant-depth QAOA on Maximum Independent Set and MaxCut respectively. As stated in [Farhi *et al.*, 2020] QAOA with depth $p < \Omega(\log n)$ can't find an independent set S with more than $0.854|S_{max}|$ nodes on a *typical* d -regular graphs. Locality together with the property of MaxCut cost function called \mathbb{Z}_2 -*symmetry* was used in [Bravyi *et al.*, 2020] to construct a *worst-case example* for which QAOA with $p < \Omega(\log n)$ can't beat the approximation ratio of the Goemans-Williamsion algorithm.

Approximation ratio of QAOA on Ising chain

We use locality to prove the simple fact that the approximation ratio of QAOA of depth p on the *ring of disagrees* is bounded by $\alpha < \frac{2p+2}{2p+3}$. This fact was proven in [Mbeng *et al.*, 2019b] using Jordan-Wigner transformation and in [Mbeng *et al.*, 2019a] using boundary conditions. An independent work [Wurtz and Love, 2021] provides a demonstration founded on ideas that are very close to ours.

The ring of disagree problem's cost function is:

$$C_r = \sum_{i=1}^n Z_i Z_{i+1} \quad (3.55)$$

where we take $Z_{n+1} \equiv Z_0$. The Hamiltonian (3.55) encodes (up to some constant factor) the MaxCut problem on an Ising chain.

Claim 3.5.2 *For the ring of disagree problem QAOA with depth p achieves the approximation ratio at most $\alpha = \frac{2p+2}{2p+3}$.*

Proof:

For simplicity, we take $n = (2p + 3) \times m$ where $m \geq 1$ is even.

Ring of disagrees is an anti-ferromagnetic Ising model on a layout $G_r = (V_r, E_r)$ that is shown on the figure 3.7a. Crucially, after p layers of QAOA the support of an observable $Z_i Z_{i+1}$

is $Q = \{u | u \in [i-p, \dots, i+p+1]\}$ (see Figure 3.6). After the index change we can set $Q = \{u | u \in [1, \dots, 2p+1]\}$. The restriction of the operator C_r on the support Q is $C_r^Q = \sum_{u=1}^{2p} Z_u Z_{u+1}$.

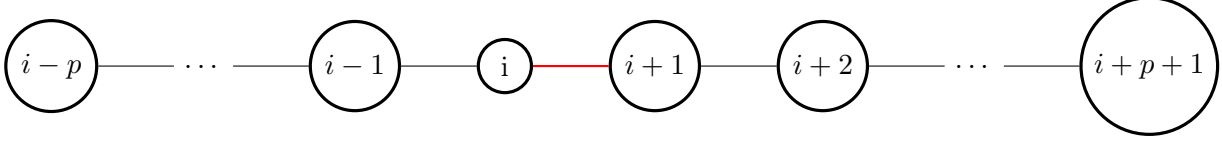


Figure 3.6: Support of the operator $Z_i Z_{i+1}$ after p layers of QAOA on the ring of disagree problem

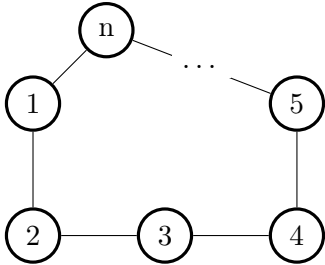
Let's arbitrary fix some $\beta, \gamma \in \mathbb{R}^p$. Due to the locality property (3.54) the expectation $\langle \psi_p^r(\beta, \gamma) | Z_i Z_{i+1} | \psi_p^r(\beta, \gamma) \rangle$ in QAOA- p state $|\psi_p^r(\beta, \gamma)\rangle$ is the same as the expectation of $Z_p Z_{p+1}$ in the state

$$|\phi_Q(\beta, \gamma)\rangle = e^{i\beta_p \sum_{q=1}^{2p+1} X_q} \dots e^{i\beta_1 \sum_{q=1}^{2p+1} X_q} e^{-i\gamma_1 \sum_{q=1}^{2p} Z_q Z_{q+1}} |+\rangle_1 \dots |+\rangle_p \quad (3.56)$$

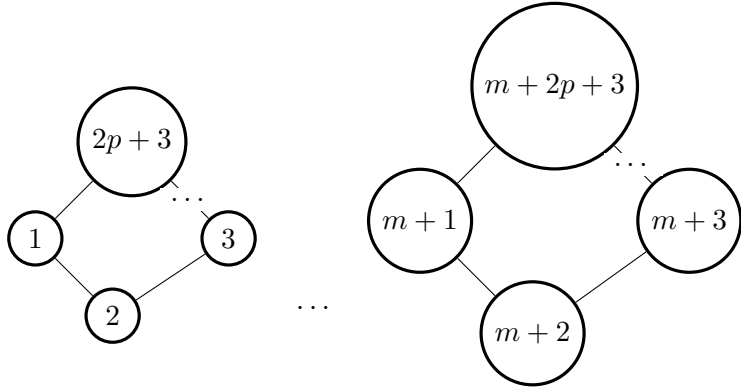
Thus the energy expectation is:

$$L_r(\beta, \gamma) = \langle \psi_p^r(\beta, \gamma) | C_r | \psi_p^r(\beta, \gamma) \rangle = n \times \langle \phi_Q(\beta, \gamma) | Z_p Z_{p+1} | \phi_Q(\beta, \gamma) \rangle \quad (3.57)$$

We consider another instance $C_d = \sum_{(u,v) \in E_d} Z_u Z_v$ where the pair interactions E_d correspond to edges of the graph $G_d = (V_d, E_d)$ made out of m disconnected cycles each of length $2p+3$. The graph G_d is shown on the figure 3.7b. We remark that $|V_r| = |V_d| = n$ and $|E_r| = |E_d| = n$.



(a) The ring of disagree graph $G_r = (V_r, E_r)$. It has $n = (2p+3) \times m$ nodes. The Hamiltonian (3.55) encodes the MaxCut problem on this graph.



(b) A graph $G_d = (V_d, E_d)$ with the same p -local structure as G_r . Each of m components has $2p+3$ nodes.

As the each small cycle in G_d has length $l > 2p+1$ the support of the observable $Z_u Z_v$, $(u, v) \in E_d$ conjugated by p layers of QAOA for C_d is exactly as on 3.6.

Using the same reasoning as before we obtain:

$$L_d(\beta, \gamma) = \langle \psi_p^d(\beta, \gamma) | C_d | \psi_p^d(\beta, \gamma) \rangle = n \times \langle \phi_Q(\beta, \gamma) | Z_p Z_{p+1} | \phi_Q(\beta, \gamma) \rangle \quad (3.58)$$

As $\beta, \gamma \in \mathbb{R}^p$ were chosen arbitrary the loss functions for two instances are equal: $L_r \equiv L_d$. As the loss function L_d is the energy expectation $\langle C_d \rangle$ in some distribution we have:

$$\min_x C_d(x) \leq \min_{\beta, \gamma} L_d(\beta, \gamma) \equiv \min_{\beta, \gamma} L_r(\beta, \gamma) \quad (3.59)$$

The ground state problem for $C = \sum_{(u,v) \in E} Z_u Z_v$ is equivalent to the MaxCut problem $\min_{x \in \{0,1\}} \text{cut}(x)$ on the layout graph $G = (V, E)$. Indeed, there is a one-to-one correspondence between solutions of two problems. The maximum cut on G_r is obtained in the partition $x^* = [0, 1, 0, 1 \dots 0, 1]$ so the ground state energy for C_r is $C_r(x^*) = \langle x^* | C_r | x^* \rangle = -n$. As graph G_d has m odd cycles its maximum cut is at most $c_{opt} = n - m$ achieved by variable assignment $x^* = [0, 1, 0, 1 \dots 0, 1]$. Therefore, the ground energy of C_d is $C_d(x^*) = \langle x^* | C_d | x^* \rangle = -n + m$.

Finally, on the ring of disagrees the QAOA with depth p has the approximation ratio:

$$\alpha_p = \frac{\min_{\beta, \gamma} L_r(\beta, \gamma)}{\min_x C_r(x)} = \frac{\min_{\beta, \gamma} L_d(\beta, \gamma)}{\min_x C_r(x)} \quad (3.60)$$

$$\geq \frac{\min_x C_d(x)}{\min_x C_r(x)} = \frac{-n + m}{-n} = \frac{2p + 2}{2p + 3} \quad (3.61)$$

■

One might expect that the integration of some global information can enhance the performance of the algorithm. Indeed, such an effect was observed on warm-start modifications where the global information is passed through the initial state [Tate *et al.*, 2020].

3.5.7 Parameter optimization

QAOA is a variational algorithm. In a manner, it doesn't define a *program* in the traditional sense but rather a *model* (commonly referred as *ansatz*) that is trained for a special purpose. Indeed, the algorithm operates a family of distributions $\mathbb{P}_{\beta, \gamma}(x)$ that can be prepared on a quantum computer. The problem to be solved determines the loss function $L(\beta, \gamma)$. The classical part has to *train the ansatz*: it has to find the values β^*, γ^* that minimize the loss function. In order for QAOA to be efficient, the training should be reasonably *fast* and it should provide an *accurate* result.

Why it is difficult to optimize parameters?

There are several obstructions that make optimization challenging. Two of them are related to the cost function structure:

- **multiple local minimas.** The loss function $L(\beta, \gamma) = \langle \psi(\beta, \gamma) | C | \psi(\beta, \gamma) \rangle$ usually has many arbitrary bad local minimas (see figure 3.8). Thus, if a gradient-based optimization method starts in a bad initial point it can end up in a very poor solution [Larkin *et al.*, 2020, Bittel and Kliesch, 2021].
- **barren plateaus.** The loss function landscape can have vast *flat zones* where the gradient vanishes, so the iterative optimizer can't improve the solution even if it is not optimal. This phenomenon was explicitly observed for random parameterized quantum circuits [McClean *et al.*, 2018]. Fortunately, low-depth QAOA applied to optimize local functions is not expected to exhibit such destructive behavior [Cerezo *et al.*, 2021b].

Besides, we usually have access only to a stochastic estimation of the loss function's value:

$$L(\beta, \gamma) \approx \frac{1}{N} \sum_{i=1}^N C(X_i) \quad (3.62)$$

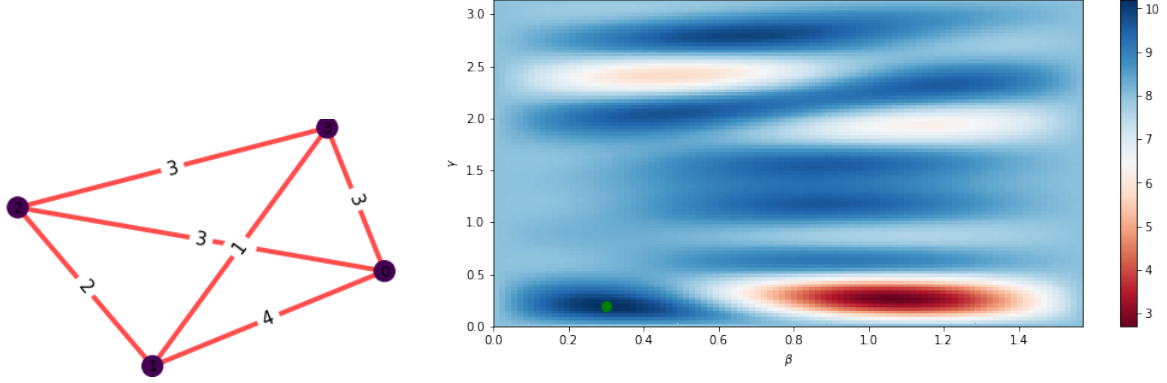


Figure 3.8: Loss function $L(\beta_1, \gamma_1)$ landscape (on the left) in QAOA₁ for MaxCut on the weighted graph (on the right)

where X_i are independently sampled from $\mathbb{P}_{\beta, \gamma}(x)$. This approach to evaluate the loss introduces a **stochastic imprecision** that linearly decreases with the number of samples N .

Moreover, the samples X_i themselves are not exactly the expected ones. In fact, in **noisy quantum hardware** measurements are not distributed precisely as $\mathbb{P}_{\beta, \gamma}(x)$ but rather according to some noisy versions $\tilde{\mathbb{P}}_{\beta, \gamma}^i(x)$ that are different for each sample i . This effect was demonstrated to have a significant impact on the viability of the stochastic estimation [Harrigan *et al.*, 2021].

As was already mentioned (see equation 3.42) when p grows the model gains more expressibility and better solves the problem. On the flip side, the training complexity drastically increases with p as any additional layer adds two extra dimensions to the optimization landscape [Shaydulin and Alexeev, 2019].

The work [Bittel and Kliesch, 2021] shows an example of NP-hard training problem for the number of parameters $p \sim n$. In this work a simple ansatz $W(\alpha_1, \dots, \alpha_n) = \prod_{u=1}^n e^{i\alpha_u Y_u}$ over n parameters is taken to solve the MaxCut problem. For this ansatz it was proven that an efficient approximation algorithm for the optimum of the loss function, if it existed, could be used to solve the MaxCut problem.

Global methods

Since the parameter landscape has multiple minima, more robust *global optimization methods* are a good choice for small p . The original QAOA paper [Farhi *et al.*, 2014a] used the *grid search*. In [Larkin *et al.*, 2020] the *Asynchronously Parallel Optimization Solver for finding Multiple Minima (APOSMM)* was shown to find good-quality parameter values [Larkin *et al.*, 2020]. In *our work* [Dalyac *et al.*, 2021] the *differential evolution* was shown to be faster than local methods.

Initialization strategies

When *local optimization methods* are used the initialization strategy is extremely important [Bittel and Kliesch, 2021].

The method can start from multiple random points, but in a difficult landscape such an approach drastically increases the total runtime. The work [Sack and Serbyn, 2021] claims that one can obtain a solution of the same quality with one point corresponding to the approximate annealing schedule as with exponentially many random point.

In [Zhou *et al.*, 2020] two initialization strategies are presented. Both reuse the optimum

values at depth $p - 1$ to initialize the search at depth p . The *INTERP* strategy uses linear interpolation, while *FOURIER* implements more complex ideas of spectrum decomposition.

As the loss function concentrates for the instances from the same distribution, one can reuse optimum values over the same families of instances. The work [Brandao *et al.*, 2018] suggests using optimum parameter values for smaller instances to initialize the search for bigger ones.

The concentration phenomenon also inspired numerous *machine learning* approaches that aim to amortize the optimization cost over several instances. The *QAOAKit* package presented in [Shaydulin *et al.*, 2021] provides pretrained optimal values for different instances of the MaxCut problem. A transfer technique suggested in [Shaydulin *et al.*, 2022] allows to reuse *QAOAKit* for *weighted* MaxCut. The work [Crooks, 2018] trains parameters *directly* on a family of instances. In [Khairy *et al.*, 2020] a *kernel density estimation* technique is adopted to learn a *generative model* of QAOA parameters that are close to the optimum.

Learning-based methods

In addition, several meta-learning techniques were suggested [Wilson *et al.*, 2021, Khairy *et al.*, 2020]. In contrast to the previously mentioned approaches that reuse the information to output good parameter values or efficient initialization points, in meta-learning techniques the *optimizer itself is trained* to efficiently guide the search to the optimum. In [Wilson *et al.*, 2021] a *Lost Short-Term Memory recurrent neural network* is trained to suggest new trial points $\beta_{t+1}, \gamma_{t+1}$ taking in input the gradient of the loss in the point β_t, γ_t . In [Khairy *et al.*, 2020] the *reinforcement learning framework* is applied to learn a *policy network* that suggests good iterative updates. Unsupervised learning strategies based on *clustering* were considered in [Moussa *et al.*, 2022].

These learning techniques can significantly reduce the QAOA runtime making it competitive with classical heuristics [Crooks, 2018].

Local methods

In brief, *global methods* are robust to local minima and *meta-learning techniques* decrease the overall runtime while improving the solution quality. Nevertheless, of-the-shelf *local optimization* algorithms are still the most commonly used ones. Due to the presence of *multiple local minimas* the performance of local heuristics is strongly dependent on the initialization strategy. Moreover, the loss function in general is *non-convex*, especially when it is estimated on noisy hardware. For instance, because of non-convexity gradient-based methods fail to converge for the *Variational Quantum Eigensolver* ansatz [Peruzzo *et al.*, 2014].

Derivative-free local heuristics such as *Nelder-Mead* simplex method [Gao and Han, 2012] and *COBYLA* [Powell, 1994] are a very popular choice for QAOA. It is partially due to a relatively small cost in terms of the number of function evaluations compared to gradient-based methods [Guerreschi and Smelyanskiy, 2017]. However, more sophisticated gradient-based method [Koczor and Benjamin, 2022] is expected to be more efficient *both* in cost and in the solution quality.

Some works use traditional gradient-based methods such as quasi-Newton gradient-based BFGS [Fletcher, 1987] and ADAM [Kingma and Ba, 2014] in the classical loop of QAOA. ADAM was reported to find better solutions than other of-the-shell methods [Tate *et al.*, 2020].

Yet special routines that are explicitly designed for quantum variational algorithms can be significantly faster. One such technique is *Quantum Analytic Descent (QAD)* [Koczor and Benjamin, 2022]. This sophisticated algorithm significantly reduces the number of calls to the quantum computer by using local classical approximations of the loss function to make "big jumps". Quantum mea-

measurements are used to evaluate coefficients of the approximation that is further optimized with several steps of the classical gradient descent. The approximation is refined in the new point and, unless the convergence criteria is met, the process restarts. This method was reported to be particularly suitable in the limited function evaluation budget context.

Gradient of the loss function

During a long time the main obstacle for gradient-based routines was the absence of a convenient way to *compute the gradient values*, even though some efforts were made in this direction [Guerreschi and Smelyanskiy, 2017]. Indeed, for the loss function value

$$L(\boldsymbol{\alpha}) = \langle \psi_0 | U^\dagger(\boldsymbol{\alpha}) O U(\boldsymbol{\alpha}) | \psi_0 \rangle \quad (3.63)$$

in most cases one have only a stochastic approximation. The situation is even more difficult for the gradient as even the stochastic option is not accessible. Indeed, the gradient of the loss has the form:

$$\frac{\partial \langle \psi_0 | U^\dagger(\boldsymbol{\alpha}) O U(\boldsymbol{\alpha}) | \psi_0 \rangle}{\partial \alpha_i} = \langle \psi_0 | U^\dagger(\boldsymbol{\alpha}) O \frac{\partial U(\boldsymbol{\alpha})}{\partial \alpha_i} | \psi_0 \rangle + \langle \psi_0 | \frac{\partial U^\dagger(\boldsymbol{\alpha})}{\partial \alpha_i} O U(\boldsymbol{\alpha}) | \psi_0 \rangle. \quad (3.64)$$

where in general the operator $U^\dagger(\boldsymbol{\alpha}) O \frac{\partial U(\boldsymbol{\alpha})}{\partial \alpha_i}$ as well as its complex conjugate is neither *unitary* nor *Hermitian*. Therefore, it can't be stochastically approximated given samples from the QAOA distribution.

The alternative is to use finite approximation:

$$\frac{\partial L}{\partial \alpha_i}(\boldsymbol{\alpha}) \approx \frac{L(\boldsymbol{\alpha} + \epsilon e_i) - L(\boldsymbol{\alpha} - \epsilon e_i)}{2\epsilon} \quad (3.65)$$

where e_i is a unit vector with only one non-zero position i .

In realistic settings the estimated values of $L(\boldsymbol{\alpha} + \epsilon e_i)$ and $L(\boldsymbol{\alpha} - \epsilon e_i)$ are imprecise due to the hardware noise and the stochastic error. These imprecisions together with the finite approximation error strongly increase the number of measurements that are necessary to obtain a sufficiently good estimation of the gradient value.

For the special case where $U(\boldsymbol{\alpha}) = e^{i\alpha\sigma}$ where σ is one-qubit Pauli gate X , Y or Z the gradient can be expressed via so-called *phase-shift rule* [Schuld *et al.*, 2019]:

$$\frac{\partial L}{\partial \alpha_i}(\dots \hat{\alpha}_i \dots) = L(\dots \hat{\alpha}_i + \frac{\pi}{4} \dots) - L(\dots \hat{\alpha}_i - \frac{\pi}{4} \dots) \quad (3.66)$$

Recently, *parameter-shift rules* appeared for more complex parameterized unitaries $U(\boldsymbol{\alpha})$ [Schuld *et al.*, 2019, Kyriienko and Elfving, 2021, Wierichs *et al.*, 2022, Izmaylov *et al.*, 2021]. Using these rules one can evaluate the gradient by querying the function values in multiple points:

$$\frac{\partial L}{\partial \alpha_{i^*}}(\hat{\boldsymbol{\alpha}}) = \sum_{k=1}^m \epsilon_k L(\boldsymbol{\alpha}^k) \quad (3.67)$$

where $\alpha_i^k = \begin{cases} \hat{\alpha}_i, & i \neq i^* \\ \hat{\alpha}_i + \phi_i, & i = i^* \end{cases}$. The *shifts* ϕ_i and coefficients ϵ_k depend on the spectrum of the parameterized gate.

We highlight that the equalities in the expressions (3.66) and (3.67) are *exact*, i.e. only stochastic error (in estimations of $L(\boldsymbol{\alpha}^k)$) remains when the rules are used to compute the gradient. This may significantly improve the performance of gradient-based optimization strategies.

A Python package *PennyLane* [Bergholm *et al.*, 2018] that was specially designed for variational quantum algorithms fully relies on these rules. We come back to parameter shift rules in chapter 9.

Which method is better?

The *speed* and the *solution quality* are the most important metrics for parameter optimization methods. Another one is the *robustness to noise* that appears in the evaluations of the loss function. Meta-learning techniques are expected to be more robust as they can potentially learn and integrate the hardware specific effects [Wilson *et al.*, 2021].

The Python package *scikit-quant* [Lavrijsen *et al.*, 2020] contains several optimization methods (*NOMAD* [Le Digabel, 2011], *ImFil* [Kelley, 2011], *SnobFit* [Huyer and Neumaier, 2008], *BOBYQA* [Powell, 2009]). The package was specially designed to have good performance in the presence of noise. This package can directly be used in hardware implementations of variational algorithms.

3.6 Recursive QAOA

One of the main reasons why QAOA may fail to find a good approximate solution is its *locality* [Farhi *et al.*, 2020, Hastings, 2019]. In other words, at low depth QAOA can't see the global structure of the instance. In order to address this drawback the work [Bravyi *et al.*, 2020] suggests to integrate QAOA in *another recursive procedure* that enforces a *non-local* treatment in the classical part. The resulting algorithm was named *Recursive QAOA (RQAOA)* and was successfully used for MaxCut [Bravyi *et al.*, 2020, Egger *et al.*, 2021] and Max-k-Cut [Bravyi *et al.*, 2022] problems.

The works [Egger *et al.*, 2021] and [Patel *et al.*, 2022] suggest techniques for further enhancement of the performance of RQAOA. In [Egger *et al.*, 2021] the optimization is *warm-started* with an additional classical preprocessing. The work [Patel *et al.*, 2022] presents an even more advanced approach that integrates RQAOA in a larger reinforcement-learning routine.

Just as Quantum Annealing and the original QAOA the recursive algorithm takes a diagonal Hamiltonian $C_{init} = \sum_{k=1}^m h_k \prod_{i \in S_k} Z_i$ on n variables in input and returns a low-energy state in output. The recursive part executes a variable elimination that stops when the instance is small enough ($n \leq \eta$) to be addressed by brute-force classical routines.

Algorithm

At each recursion step the variable elimination routine considers a reduced instance C_t over $n - t$ variables V_t . Initially, C_0 is taken as $C_0 = C_{init}$, $V_0 = [1, \dots, n]$. The choice of the leaving variable is assisted by the QAOA routine for the Hamiltonian C_t . The procedure consists of the following steps:

1. Consider the family of QAOA states $|\psi_t(\beta, \gamma)\rangle$ for the instance C^t over variables V_t
2. Find the optimal parameter values β_t^* and γ_t^* , i.e. solve the loss minimization problem

$$\beta_t^*, \gamma_t^* = \underset{\beta, \gamma}{\operatorname{argmin}} \langle \psi_t(\beta, \gamma) | C_t | \psi_t(\beta, \gamma) \rangle \quad (3.68)$$

3. For each variable $u \in V_t$ compute the expectation of the observables Z_u :

$$E_u = \langle \psi_t(\beta, \gamma) | Z_u | \psi_t(\beta, \gamma) \rangle \quad (3.69)$$

Find the variable u^* that has the biggest absolute expectation value:

$$u^* = \operatorname{argmax} |E_u| \quad (3.70)$$

4. Do the same for each pair of variables $w, v \in V_t$, i.e. compute:

$$E_{w,v} = \langle \psi_t(\beta, \gamma) | Z_w Z_v | \psi_t(\beta, \gamma) \rangle \quad (3.71)$$

and find the pair

$$p^* = (w^*, v^*) = \operatorname{argmax} |E_{w,v}| \quad (3.72)$$

5. If $|E_{u^*}| \geq |E_{w^*,v^*}|$ fix the value of the variable u^* to $\operatorname{val}(u^*) = \sigma(E_{u^*})$ and remove the variable u^* : $V_{t+1} = V_t \setminus \{u^*\}$. We denote by $\sigma(\cdot)$ the sign function: $\sigma(x) = \begin{cases} -1, & x \leq 0 \\ 1, & x > 0 \end{cases}$.
6. Otherwise fix the correlation of the variables w^*, v^* to $\operatorname{val}(w^*)\operatorname{val}(v^*) = \sigma(E_{w^*,v^*})$ and remove the variable w^* : $V_{t+1} = V_t \setminus \{w^*\}$.
7. Repeat until the number of variables is small: $|V_t| \leq \eta$ and then find the ground state of C_t with a brute-force routine.

How to modify the Hamiltonian

When the routine fixes a value of a variable or a correlation it modifies the Hamiltonian $C_t \rightarrow C_{t+1}$. The new instance C_{t+1} is designed to be equivalent to its predecessor on solutions that satisfy the fixed constraint. If C_t is an Ising model the modification is straightforward. Let

$$C_t = \sum_{u \in V_t} h_u Z_u + \sum_{u,v \in V_t} h_{u,v} Z_u Z_v \quad (3.73)$$

Fixing $\operatorname{val}(u^*) = \sigma$ for $\sigma \in \{-1, 1\}$ implies $Z_{u^*} = \sigma$. Therefore, the new Hamiltonian C_{t+1} that is equivalent to C_t on variable assignments satisfying $\operatorname{val}(u^*) = \sigma$ is

$$C_{t+1} = \sum_{u \in V_t \setminus \{u^*\}} h_u Z_u + \sum_{u,v \in V_t \setminus \{u^*\}} h_{u,v} Z_u Z_v + \sigma \sum_{v \in V_t \setminus \{u^*\}} h_{u^*,v} Z_v \quad (3.74)$$

Similarly, on assignments that respect $\operatorname{val}(w^*)\operatorname{val}(v^*) = \sigma$, $\sigma \in \{-1, 1\}$ we have $Z_{w^*} Z_{v^*} = \sigma$ or $Z_{w^*} = \sigma Z_{v^*}$ (as $Z^2 \equiv I$). The Hamiltonian C_{t+1} is:

$$\begin{aligned} C_{t+1} = & \sum_{u \in V_t \setminus \{w^*, v^*\}} h_u Z_u + (h_{v^*} + \sigma h_{w^*}) Z_{v^*} \\ & + \sum_{u,v \in V_t \setminus \{w^*, v^*\}} h_{u,v} Z_u Z_v + \sum_{v \in V_t \setminus \{w^*\}} (h_{v^*,v} + \sigma h_{v^*,v}) Z_{v^*} Z_v \end{aligned} \quad (3.75)$$

Motivation of the variable elimination routine

To make the intuition behind the variable elimination clear let's consider an observable O such that $O|x\rangle = \pm r|x\rangle$ on vectors $|x\rangle$ from the computational basis. If in some state $|\psi\rangle$ the expectation $\langle\psi|O|\psi\rangle \approx +r$ one can conclude that the probability distribution $\mathbb{P}(x) = |\langle x|\psi\rangle|^2$ concentrates on states $|x\rangle$ that corresponds to the eigenvalue $+r$.

QAOA distribution is chosen to favor solutions corresponding to low values of the objective function. Therefore, if in such distribution the mean value of an observable $Z_u : Z_u|x\rangle = \pm 1|x\rangle$ is close to $\sigma = 1$ one can expect that the variable takes the value 1 in most low-energy states. So it makes sense to set $val(u) = 1$.

Classical simulation

A particularly pertinent fact about RQAOA is that for low depths it can be fully simulated on a classical computer [Bravyi *et al.*, 2022, Egger *et al.*, 2021]. Indeed, for $p = 1$ all expectations $\langle\psi_1(\beta, \gamma)|Z_u|\psi_1(\beta, \gamma)\rangle$, $\langle\psi_1(\beta, \gamma)|Z_u Z_v|\psi_1(\beta, \gamma)\rangle$ and $\langle\psi_1(\beta, \gamma)|C|\psi_1(\beta, \gamma)\rangle$ can be evaluated using analytical formulas, for example our formula (3.37).

We recall that for the original QAOA even at $p = 1$ an efficient simulation algorithm would make the polynomial hierarchy collapse [Farhi and Harrow, 2016]. Thereby, computing the *mean value of an operator* in the QAOA state (at least for low depths) is fundamentally *easier* in terms of the complexity scaling on the instance size than to obtain a *sample* from the corresponding QAOA distribution. In particular, for $p = 1$ we have a polynomial classical algorithm that returns the values $\langle\psi_1(\beta, \gamma)|O|\psi_1(\beta, \gamma)\rangle$ for $O \in \{Z_u, Z_u Z_v, C\}$ but, unless the polynomial hierarchy collapses, there is no classical algorithm that returns $X \sim \mathbb{P}_1(\beta, \gamma)$ in polynomial time.

Chapter 4

Smart Scheduling

Due to the rapid development of electric mobility, problems related to vehicle charging gain in importance. In this chapter we consider a *scheduling problem*, the goal is to assign a set of charges to different charging stations.

Scheduling problems are extremely important in industrial applications. Essentially, a *schedule* is an ordered assignment of n jobs to m machines. At every moment a job can be processed on only one machine and each machine can process no more than one job.

There are many variants of scheduling problems with diverse *machine environments*, *job characteristics* and *optimality criteria*. Possible machine environments include *single* processor setup as well as multiple processor setups. In the latter case machines can be *parallel*, *uniform* or *unrelated*. On the other side, jobs themselves have different parameters. For instance, they can be preemptive or not. There may also be a partial order defined on the set of jobs. Such order usually models a precedence requirement where a job can be executed only after some preparation tasks are completed. Diverse problem settings and possible solution techniques are presented in the survey [Graham *et al.*, 1979]. The survey also introduces the widely adopted classification for scheduling problems in the form $\alpha|\beta|\gamma$. In this expression α relates to the machine settings, β - to diverse job characteristics and γ - to the optimality criteria.

4.1 Problem statement

In the smart scheduling problem we are given a set of charge demands J , $|J| = n$. Each job $j \in J$ has a fixed *processing time* $t_j \in \mathbb{R}^+$ and *priority* $w_j \in \mathbb{R}^+$. Jobs have to be assigned to a set M of m *parallel (identical)* charging stations. Identical stations imply that the processing time for a job is always t_j , i.e it doesn't depend on a specific terminal it is assigned to. We assume that jobs are *non-preemptive*. In other words, once a processor $i \in M$ starts to treat a job the execution can't be interrupted until the job is completed.

A given schedule S determines the completion time C_j for each job $j \in J$. Intuitively, the completion time equals the processing time t_j plus the time that the job has to wait before being executed:

$$C_j = W_j + t_j = \left(\sum_{\substack{i \in J \\ i \prec_S j}} t_i \right) + t_j \quad (4.1)$$

where the condition $i \prec_S j$ is true if the schedule S assigns the job i to the same machine as j and i is executed before j .



Figure 4.1: In realistic setting *priority* can relate to the importance of a particular vehicle. For example, for a manager of a municipal car park it can be less desirable to delay a police car than a bus.

We are interested in a schedule that minimizes the total weighted completion time: $\sum_j w_j C_j$. In the classification of [Graham *et al.*, 1979] the problem can be written as $P||\sum_j w_j C_j$ if the *number of stations* is a part of the input and $Pm||\sum_j w_j C_j$ if it is fixed.

4.1.1 Complexity and approximations

In some particular cases this scheduling problem can be efficiently solved. For instance, in a single station case $1||\sum_j w_j C_j$ a polynomial algorithm directly follows from the *Smith's Rule* [Smith, 1956]. According to this rule, it suffices to process jobs in a non-increasing order given by ratios w_i/t_i . For multiple station case the problem can also be solved in $O(n \log n)$ if all priorities w_j are equal ($P||\sum_j C_j$).

However, for general positive priorities and for fixed $m \geq 2$ the problem is NP-hard in the ordinary sense [Skutella and Woeginger, 2000]. Moreover, if the number of stations m is a part of the input the scheduling problem $P||\sum_j w_j C_j$ is *NP-hard in the strong sense* [Garey and Johnson, 1990]. Yet, even if the optimal solution is hard to compute a relatively efficient approximation is achievable. The work [Skutella and Woeginger, 2000] presents a *polynomial-time approximation scheme (PTAS)* that, given an arbitrarily fixed approximation ratio $\alpha = 1 + \epsilon$ for $\epsilon > 0$ computes the approximate optimum in time polynomial in n and m . The suggested procedure has an *exponential* complexity on the precision $1/\epsilon$. We recall that a family of $1 + \epsilon$ approximation algorithms with complexity growing *polynomially* in $1/\epsilon$ is called *fully polynomial-time approximation scheme (FPTAS)*. As problems that are NP-hard in a strong sense are not expected to have a FPTAS, the PTAS for $P||\sum_j w_j C_j$ can't be significantly outperformed.

On the other hand, for a fixed number of stations m the problem $Pm||\sum_j w_j C_j$ is pseudopolynomial⁶ and admits a FPTAS [Sahni, 1976, Woeginger, 2001]. These results rely on the dynamic programming formulating of the scheduling problem.

The dynamic program maintains a set of partial solutions. Partial solutions are iteratively extended to full solutions. The dynamic algorithm finds an exact optimum but may have an exponential runtime. In order to guarantee a polynomial-time execution the work [Sahni, 1976] suggests an approximate scheme based on *rounding of the input data*. For $m = 2$ and arbitrary $\epsilon > 0$ the complexity of the scheme is $O(n^2/\epsilon)$.

⁶It can be solved with an algorithm in a time that is polynomial in the numeric values of its data $O(\text{poly}(\sum_j w_j + \sum_j t_j))$ but not on the length of their binary encodings.

Alternatively, the work [Woeginger, 2001] obtains an approximation scheme by performing a *trimming of the partial solution space* at each step of the dynamic program. This approach can be applied to a wide range of problems that admit a dynamic programming formulation satisfying certain properties (are *DP-benevolent*). The *trimming technique* has also inspired a class of polynomial-time *evolutionary algorithms* with guaranteed approximation ratios [Doerr *et al.*, 2011].

Although it cannot be expected that quantum algorithms outperform the specific FPTAS on the given scheduling problem, it is nevertheless interesting to analyze what performance quantum heuristics can achieve.

4.1.2 Max-m-Cut reformulation

In order to apply quantum algorithms to a scheduling problem one has to find a corresponding QUBO formulation. One QUBO formulation for $P2||\sum_j w_j C_j$ was suggested in [Alidaee *et al.*, 1994]. In *our work* [Dalyac *et al.*, 2021] we consider the mapping [Skutella, 1998] that associates to each scheduling instance $Pm||\sum_j w_j C_j$ an instance of the weighted Max-m-Cut problem.

More precisely, we define a complete weighted graph $G = (V, E)$ where the nodes V correspond to charging jobs J . To each pair of nodes (u, v) , $u \in V, v \in V$ is associated a weight $e_{u,v} = \min\{w_u t_v, w_v t_u\}$. We aim to find a partition $V = V_1 \sqcup \dots \sqcup V_m$ of the set of nodes such that the total weights of edges that are *cut* by the partition is maximal. An edge (u, v) is *cut* if its endpoints belong to different members of the partition: $u \in V_i, v \in V_j, i \neq j$. The Max-m-Cut problem can be written as:

$$\max_{V=V_1 \sqcup \dots \sqcup V_m} \text{cut}(V) = \sum_{1 \leq s < s' \leq m} \sum_{u \in V_s} \sum_{v \in V_{s'}} e_{u,v} \quad (4.2)$$

A solution for the Max-m-Cut problem (4.2) defines an assignment of the charge jobs to different processors. Jobs that belong to the same partition V_s are assigned to the same station $s \in M$. Then for each station a locally optimal schedule is obtained using the *Smith's Rule*. An example of a scheduling problem on 3 stations and the corresponding instance of Max-3-Cut is shown on the figure 4.2.

In the rest of the section we explain why this reformulation is indeed correct.

For the total completion time in the scheduling problem we have:

$$\sum_j w_j C_j = \underbrace{\sum_j w_j W_j}_{\mathcal{W}} + \underbrace{\sum_j w_j t_j}_{\mathcal{E}} \quad (4.3)$$

where $\mathcal{E} = \text{const}$ that doesn't depend on a particular schedule. The waiting time is $W_j = \sum_{i \in P_j} t_i$ where $P_j \subset J$ is the set of jobs scheduled before j on the same station.

If two jobs i and j are assigned to the same station $s \in M$ one of them has to wait for another to be completed. According to the Smith Rule, if $w_i/t_i > w_j/t_j$ (or, equivalently, $w_i t_j > w_j t_i$) the job j will be scheduled after i . Therefore, the job $i \in P_j$ and the value $w_j t_i$ will contribute to component \mathcal{W} in the sum (4.3). In other words, if i and j are assigned to the same part V_s the weight of the edge $e_{u,v}$ appears in the total weighted completion time:

$$\begin{aligned} \sum_j w_j W_j &= \sum_{u,v \in V} e_{u,v} I(\exists s : i \in V_s, j \in V_s) \\ &= \sum_{u,v \in V} e_{u,v} (1 - I(u \in V_s, v \in V_{s'}, s \neq s')) \end{aligned}$$

Input:

$$V = \{a, b, c, d, e\}$$

$$T = [1, 2, 2, 1, 1]$$

$$W = [1, 1, 1, 1, 5]$$

Output:

st.1 : e

st.2 : b, a

st.3 : c, d

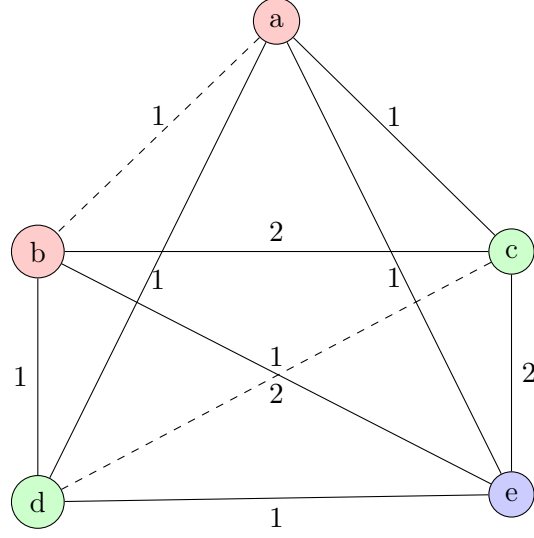


Figure 4.2: Example of *smart-charging* instance for $m = 3$ and the corresponding instance of the Max-3-Cut problem. An optimal partition is given by colors.

$$\equiv \sum_{u,v \in V} e_{u,v} - \text{cut}(\mathcal{V}) \quad (4.4)$$

The last expression (4.4) shows that a partition that maximizes $\text{cut}(\mathcal{V})$ correspond to a schedule that minimizes the total weighted completion time $\mathcal{W} + \mathcal{E}$. Therefore, an algorithm with an approximation ratio α_{MC} for Max- m -Cut can be adapted to solve the scheduling problem with the ratio $\alpha_S = \alpha_{MC} + m(1 - \alpha_{MC})$ [Skutella, 1998]. For instance, when the Max-2-Cut subproblem is addressed with Goemans-Williamson algorithm the completion time of the approximate optimum schedule is of factor $\alpha_S = 1.122$ from the optimum value.

The finite ratio $\alpha_S = 1.122$ can't compete with the solution quality of FPTAS for the scheduling problem. However, if Max- m -Cut is approximately solved with *an efficient heuristic* (for example a *quantum heuristic*) the runtime advantage can make the Max- m -Cut model attractive for practical use. Besides, a qualification of quantum heuristics on easy-to-approximate instances allows to better understand their computational performance.

4.2 Max- m -Cut

In the Max- m -Cut problem one aims to color a set of nodes of a weighted graph $G = (V, E)$ in a way that maximizes the total weight of *properly colored edges* (see the equation 4.2). A coloring $\mathcal{C} : V \rightarrow [1, \dots, m]$ defines a partition $\mathcal{V} = \sqcup_{c=1}^m V_c$ where each part corresponds to one color: $V_c = \{v \in V \mid \mathcal{C}(v) = c\}$.

4.2.1 The case $m = 2$

The case $m = 2$ is by far the most studied one both in classical and in quantum computer science. The Max-2-Cut problem is also referred as MaxCut or Max-2-XORSAT. The problem is NP-hard. Moreover, if $P \neq NP$ there is no algorithm for unweighted MaxCut with approximation

ratio better than $0.941\dots$ [Arora *et al.*, 1998]. It was shown in [Crescenzi *et al.*, 2001] that a weighted version of the problem has essentially the same hardness as the unweighted one.

The best proven approximation ratio for MaxCut is $\alpha_{GW} \geq 0.87856$. This ratio is achieved by the algorithm suggested by Goemans and Williamson [Goemans and Williamson, 1995]. In the Goemans-Williamson algorithm a SDP-relaxation of the cut problem is solved. The discrete partition is extracted from the continuous solution of the SDP by a special randomized rounding procedure. The bound $\alpha_{GW} \geq 0.87856$ was proven to be tight [Karloff, 1999], i.e. there exist instances for which the algorithm achieves *exactly* the ratio 0.87856. Moreover, under the *Unique Game Conjecture* [Khot, 2002] the ratio 0.87856 is optimal [Khot *et al.*, 2007].

Unique Game Conjecture

The *Unique Game* is a one round game with one verifier and two provers that can't communicate. A verifier sends two inputs s and t to the first and the second prover respectively. Each prover sends back one answer from the alphabet \mathcal{A} (also called domain) of size $k = |\mathcal{A}|$. The verifier receives $a, b \in \mathcal{A}$ and *accepts* the answers if $\sigma_{st}(a) = b$ where σ_{st} is some permutation on the set \mathcal{A} . A value ν of the game is a maximum success probability that provers can achieve.

Roughly speaking, the *Unique Game Conjecture (UGC)* states that it is NP-hard to distinguish between almost satisfiable and almost unsatisfiable unique games:

Conjecture: For any $\epsilon > 0$ and $\delta > 0$ there exists $k = k(\epsilon, \delta)$ such that for a Unique Game with answers from the domain of cardinality $k(\epsilon, \delta)$ it is NP-hard to say if its value ν is $\nu > 1 - \epsilon$ or $\nu < \delta$.

Despite significant efforts, the UGC remains unproven. An peculiar results about the validity of the conjecture for quantum Unique Games was reported in [Kempe *et al.*, 2007]. In the quantum version of the Unique Game problem provers, while still not allowed to communicate, share an *entangled state*.

Entanglement is quantum resource that sometimes leads to an increase in the success probability of games. For instance, for a famous CHSH game classical non-entangled provers achieve a value of 75% while an entangled strategy succeed in 85% of cases [Bell, 1964].

The value of a quantum Unique Game problem is usually referred as *entangled value* ν_e . It turns out that, contrary to the value of classical Unique Games, the entangled value can be efficiently approximated by a semidefinite program [Kempe *et al.*, 2007]. We recall that the approximation hardness for MaxCut for ratios $\alpha > \alpha_{GW}$ follows from the reduction from the classical Unique Game problem and the conjecture about the *classical value* ν . Therefore, an efficient approximation for the *entangled value* ν_e *doesn't* invalidate the inapproximability bound.

Other than the Goemans-Williamson algorithm, there exist multiple polynomial *heuristics* solving the MaxCut problem. For instance, MaxCut can be straightforwardly mapped to a QUBO and solved with general QUBO heuristics. It was shown that the *Tabu Search* have a good performance on QUBO formulation of MaxCut [Kochenberger *et al.*, 2013].

Moreover, some restricted classes of instances can be solved to optimality or to any desired approximation ratio $1 - \epsilon$ in polynomial time. This is the case for planar graphs [Deza and Laurent, 1994]. For k -regular *large girth* graphs a near-optimal solution may be computed in $O(nkl)$ under some widely-believed conjecture [Alaoui *et al.*, 2021].

Regular graphs often appear in experimental benchmarks for quantum heuristics [Farhi *et al.*, 2014a, Basso *et al.*, 2021]. If this is the case, the classical competitor has to be carefully chosen. For

instance, for graphs with maximum degree 3 that often appear in QAOA literature an SDP-based algorithm achieves the approximation ratio $0.9326 > \alpha_{GW}$ [Halperin *et al.*, 2004].

Quantum heuristics for optimization are commonly tested on the MaxCut problem. For low-depth QAOA several theoretical guarantees were established [Hadfield, 2018, Basso *et al.*, 2021, Wurtz and Love, 2021]. Yet, these guarantees don't demonstrate a proven quantum advantage. Moreover, there exists a family of instances such that QAOA with depth $p = \Omega(\log n)$ has worse approximation ratio than the Goemans-Williamson algorithm.

Generally, the performance of QAOA heavily depends on the considered instances [Larkin *et al.*, 2020]. It was observed that QAOA performs better on graphs with many small cycles [Wurtz and Love, 2021]. A machine-learning based approach that predicts the performance of QAOA on different instances of MaxCut was presented in [Moussa *et al.*, 2020].

An instance of the Smart Scheduling problem maps to an instance of weighted MaxCut on a *complete weighted graph*. Previously, QAOA was applied to MaxCut on *complete graphs* issued from the Sherrington-Kirkpatrick model [Farhi *et al.*, 2022].

4.2.2 Complexity and approximations for $m > 2$

For general $m > 2$ the Max-m-Cut problem is known to be NP-hard to approximate with a factor $\alpha > 1 - 1/(132m)$ [Kann *et al.*, 1997]. A naive heuristic that randomly choses a color for each node has an expected performance of $\alpha_R = 1 - 1/m$.

The work [Frieze and Jerrum, 1997] presented an algorithm that outperforms the randomized heuristic. The approximation ratio α_m of this algorithm asymptotically scales as:

$$\alpha_m \sim \left(1 - \frac{1}{m}\right) + \frac{2 \ln m}{m^2} \quad (4.5)$$

If the *Unique Game Conjecture* holds, the asymptotic scaling (4.5) is essentially optimal. In other words, if there were an algorithm with approximation ratio $\alpha_m > 1 - \frac{1}{m} + \frac{2 \ln m}{m^2}$ the Unique Game problem would be easy [Khot *et al.*, 2007].

4.3 QAOA for Smart Scheduling

In previous sections we have described the mapping of the smart scheduling problem to the Max-m-Cut problem. In the current section we provide a detailed protocol that allows to run QAOA on the Max-m-Cut instances issued from this mapping. Such protocol should specify the building components of QAOA, i.e. the families $U_P(\gamma)$ and $U_M(\beta)$ of phase and mixing operators respectively. In addition, it should contain a strategy for parameter optimization. In this section we present our protocol (previously reported in *our work* [Dalyac *et al.*, 2021]) together with its performance evaluation.

4.3.1 Encodings

A natural formulation of the Max-m-Cut problem (sometimes referred as *Max-m-colorable sub-graph* problem) employs integer variables $x_i \in [1, \dots, m]$. The variables correspond to colors assigned to nodes. For a graph with n nodes we take n integer variables x_i . Then the problem can be formulated as:

$$\max_{x \in [1, \dots, m]^n} \sum_{(u,v) \in E} e_{u,v} I(x_u \neq x_v) \quad (4.6)$$

In theory, the Max-m-Cut problem can be directly encoded using m -level physical systems called *qudits*. QAOA and RQAOA implementations on qudits were applied to the Max-3-Cut problem in [Bravyi *et al.*, 2022]. According to presented numerical results, both algorithms outperform the classical *Newman method* while RQAOA achieves better results than the traditional QAOA.

Unary encoding

The work [Wang *et al.*, 2020] suggests to map an integer variable $x_i \in [1, \dots, m]$ to a sequence of m binary variables $\mathbf{u}_i = [u_i^1, \dots, u_i^m]$. If an initial variable has the value $x_i = k$ all bits in \mathbf{u}_i are set to zero except u_k that is set to one. Such mapping is called *unary encoding*, it constitutes a state-of-the-art approach to deal with bounded integer variables [Hadfield *et al.*, 2019].

A sequence on m bits is a valid codeword if the following equality holds:

$$\sum_{1 \leq j \leq m} u_i^j = 1 \quad (4.7)$$

The energy of the system is given by the Hamiltonian:

$$C = \sum_{(u,v) \in E} e_{u,v} C_{u,v} \quad (4.8)$$

where for every edge $(u, v) \in E$ the operator $C_{u,v}$ can be as the following Ising model:

$$C_{u,v} = \frac{1}{4} \sum_{j \in m} (1 + Z_u^j Z_v^j) \quad (4.9)$$

Acting on a bitstring $|\mathbf{x}\rangle$ that is a valid codeword (i.e. \mathbf{x} satisfies 4.7) the operator $C_{u,v}$ takes the value 1 if the edge (u, v) is cut and 0 otherwise.

For the Ising energy operator C the circuit implementing the energy evolution $U_C(\gamma) = e^{i\gamma C}$ is straightforward. The *genuine* implementation challenge emerges from the *validity constraints* (4.7).

One approach is to *penalize* unfeasible solutions in the objective function with the term $\lambda(1 - \sum_{1 \leq j \leq m} u_i^j)^2$. For λ that is big enough unfeasible solutions will be significantly suboptimal. However, the penalization may impact the performance of the quantum heuristic by "flattening" the energy landscape around feasible solutions.

Another approach is to modify the initial state $|\psi_0\rangle$ and the mixing family $U_M(\beta)$ in order to maintain the evolution in a subspace of valid codewords. This approach was developed in [Wang *et al.*, 2020] and [Hadfield *et al.*, 2019]. It proposes to replace the default operator B with an XY-Hamiltonian:

$$B = \sum_{u \in V} B_u \quad (4.10)$$

$$B_u = \sum_{1 \leq i < j \leq m} X_i X_j + Y_i Y_j \quad (4.11)$$

The unary encoding is demanding in number of qubits: $n \times m$ binary variables are necessary for a graph with n nodes. Furthermore, there is no obvious way to compile QAOA components into a sequence of elementary gates. Indeed, the unitary $e^{i\beta B_u}$ with an all-to-all connections

can't be decomposed in a simple product of multiqubit Pauli rotations. As the terms $\sigma_i \sigma_{i+1}$ and $\sigma_{i-1} \sigma_i$ don't commute (σ is Pauli-X or Pauli-Y gate), the compilation issue remains open even for a simpler *ring mixer*:

$$B_u = \sum_{1 \leq i \leq m} X_i X_{i+1} + Y_i Y_{i+1} \quad (4.12)$$

Binary encoding

In *our work* [Dalyac *et al.*, 2021] we suggested to use a more resource-efficient *binary encoding*. For $m = 2^l$ each integer variable $x_i \in [0, \dots, m-1]$ is mapped to a sequence of $l = \log_2 m$ binary variables. A bitstring $\mathbf{b}_i = [b_0, \dots, b_{l-1}]$ corresponds to the integer $x_i = \sum_{0 \leq j \leq l-1} 2^j b_j$. In this encoding every binary string of length l is a valid codeword. The initial state and mixing can be taken as in constraint-free QAOA:

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \quad (4.13)$$

$$U_M(\beta) = e^{i\beta B}, \quad B = \sum_{v \in V} \sum_{0 \leq i \leq l-1} X_v^i \quad (4.14)$$

In binary encoding the energy can be written as a sum over edge set:

$$C = \sum_{(u,v) \in E} e_{u,v} C_{u,v} \quad (4.15)$$

$$(4.16)$$

where each edge is represented by the operator:

$$C_{u,v} = I(\mathcal{C}(u) \equiv \mathcal{C}(v)) = \prod_{0 \leq i < l} \frac{(1 - Z_u^i Z_v^i)}{2} \quad (4.17)$$

The expression (4.17) contains higher order monomials $M_Q = \prod_{i \in Q} Z_i$, $|Q| > 2$. Thus, it is beyond the realm of a pure Ising model. Nevertheless, the phase operator

$$U_C(\gamma) = e^{i\gamma C} = \prod_Q e^{i\gamma c_Q M_Q} \quad (4.18)$$

can be easily compiled to a quantum circuit. Indeed, for the monomial $M_Q = \prod_{i \in Q} Z_i$ of degree $|Q| = d$ the circuit for $e^{i\gamma c_Q M_Q}$ is given by:

$$\begin{array}{c}
 q_1 \text{ --- } \bullet \text{ --- } \text{---} \text{---} \text{---} \text{---} \bullet \text{ ---} \\
 q_2 \text{ --- } \oplus \text{ ---} \quad \dots \quad \text{---} \oplus \text{ ---} \\
 \quad \quad \quad \dots \\
 q_{d-1} \text{ --- } \bullet \text{ ---} \text{---} \text{---} \text{---} \bullet \text{ ---} \\
 q_d \text{ --- } \oplus \text{ ---} \boxed{R_Z(c_{u,v}\gamma)} \text{ ---} \oplus \text{ ---}
 \end{array} \quad (4.19)$$

where $Q = \{q_1, \dots, q_d\}$. The decomposition (4.19) contains $O(d)$ CNOTS.

Due to the presence of CNOTS the binary encoding, while using less qubits ($n \times \log_2 m$), leads to circuits of higher depth. The tradeoff between the number of qubits and the depth of the circuit

for different encodings of the *Traveling Salesman problem* was discussed in [Glos *et al.*, 2022]. Deeper circuits are more affected by noise. Nevertheless, in certain settings (for instance for quantum computers with small number of qubits) a higher-degree formulation may be beneficial [Glos *et al.*, 2022].

Our approach allows to tackle problems with m that is power of 2. In [Fuchs *et al.*, 2020] the binary encoding was extended to deal with arbitrary natural $m \in \mathbb{N}$. A sequence of $l = \lceil \log_2 m \rceil$ qubits is used to encode the color of a node. Bitstrings that are decoded in "overlong" integers $x \in [m, \dots, 2^l - 1]$ are "stitched" to represent the same color. This extension removes the requirement $m \equiv 2^l$. The standard initial state (4.13) and mixing operators (4.14) are still accurate in this scheme. On the down side, stitching leads to sophisticated *phase operators* that compile to circuits with an exponential amount $O(e^{(2^l - m)})$ of CNOT gates.

4.4 Numerical results

4.4.1 Experimental setup

We analyzed the performance of QAOA on graphs corresponding to the Smart Scheduling problem. Instances were generated out of a real-world dataset containing about 2250 loads performed during May 2017 on identical charging points of the Belib's network. Considered Belib load stations are located in Paris, France [Bel, 2017]. The dataset provided realistic charge durations $t_j \in \mathbb{R}^+$ but not the priorities. We considered integer priorities $p_j \in \mathbb{N}$. The values of p_j were independently sampled from the Poisson distribution:

$$\mathbb{P}(p_i = k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad (4.20)$$

Reported numerical results correspond to an ideal (noise-free) QAOA. For $p = 1$ QAOA was simulated on a regular laptop. For higher depths $p > 1$ the computations were performed on the *Quantum Learning Machine* provided by *Atos* [QLM, 2022].

For each number of nodes ranging from 6 to 250 we generated $N = 100$ different instances of the $P2||\sum_j w_j C_j$ problem.

We decided to restrict the numerical simulations to the case $m = 2$ for multiple practical reasons.

Firstly, even on the QLM supercomputer the runtime of the brute-force circuit simulation is prohibitive for $n \geq 30$ binary variables. Therefore, even for $m = 3$ we can't simulate QAOA on instances containing more than 10 jobs. Unfortunately, numerical results on such small instances can't lead to any sound conclusions about the scaling of the algorithm [McClean *et al.*, 2021].

In addition, we were able to derive an analytical formula for the loss function for the weighted MaxCut (3.37). The analytical formula allows to evaluate the approximation ratio of QAOA at $p = 1$ for relatively big instances. For instance, we were able to treat the instances with $n_{max} = 250$ nodes.

4.4.2 Establishing the protocol for QAOA

Initialization strategy

The work [Brandao *et al.*, 2018] reports the concentration phenomena for QAOA on unweighted MaxCut. Roughly speaking, according to this phenomenon QAOA parameters have similar

values on instances that are sampled from the same distribution. We have observed the same behavior for the optimal parameters on instances of the scheduling problem (see fig. 4.3).

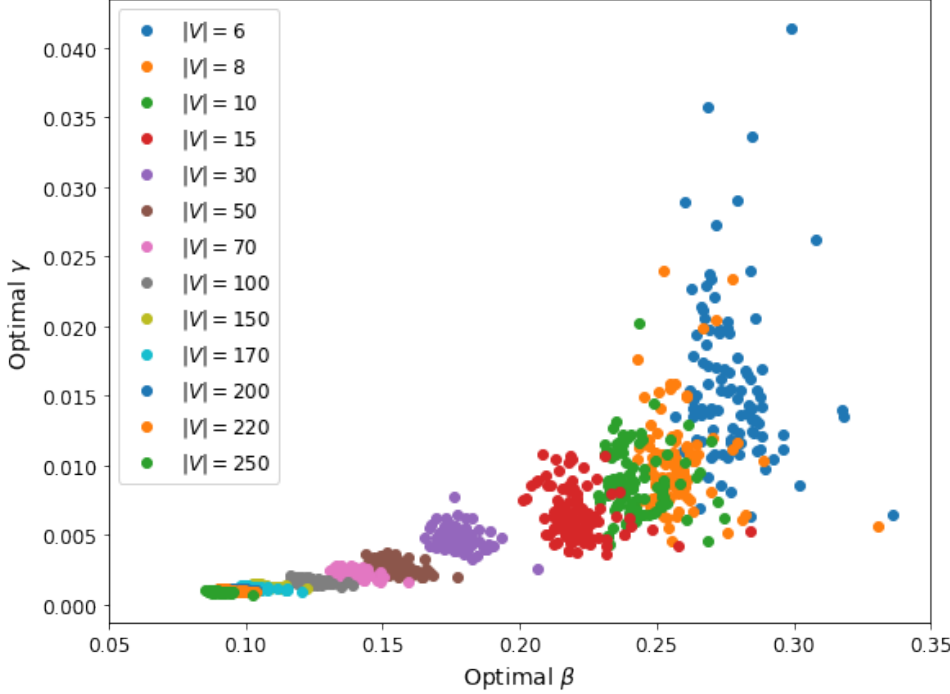


Figure 4.3: Optimal parameters β_1 and γ_1 for QAOA with depth $p = 1$ on instances of Smart Scheduling of sizes $n \in [6, 8, 10, 15, 30, 50, 70, 100, 150, 170, 200, 220, 250]$. Optimal values are computed with *grid search*. We observe that for the instances of the same size n the optimal parameters take values that are close. Concentration is more important for bigger instances. Moreover, the absolute values of parameters get closer to zero with n , i.e. *QAOA approaches the randomized algorithm*.

Having confirmed the expectation about parameter concentration, we decided to use this phenomenon to predict good initialization points for different sizes n . Notably, in our experiments we initialized local search routines in a barycenter of previously seen parameter values $(\beta_{init}^n, \gamma_{init}^n) = (\sum \beta_i^n / N, \sum \gamma_i^n / N)$. We have observed that starting from such point, local methods were able to converge to the global optimum on *all instances* from the dataset.

For higher depths $p > 1$ we used the *INTERP* approach presented in [Zhou *et al.*, 2020]. The *INTERP* strategy relies on the assumption that optimal parameters at depth $p + 1$ are close to optimal parameters at depth p . More precisely, the strategy suggests to initialize a local search at depth $p + 1$ in the point $\beta^i = [\beta_1^i, \dots, \beta_{p+1}^i]$, $\gamma^i = [\gamma_1^i, \dots, \gamma_{p+1}^i]$ such that

$$\beta_j^i = \frac{j-1}{p} \bar{\beta}_{j-1} + \frac{p-i+1}{p} \bar{\beta}_j \quad (4.21)$$

$$\gamma_j^i = \frac{j-1}{p} \bar{\gamma}_{j-1} + \frac{p-i+1}{p} \bar{\gamma}_j \quad (4.22)$$

where $[\bar{\beta}_1, \dots, \bar{\beta}_p]$ and $[\bar{\gamma}_1, \dots, \bar{\gamma}_p]$ are *optimal parameters* for QAOA at depth p . The auxiliary values $\bar{\gamma}_0, \bar{\gamma}_{p+1}, \bar{\beta}_0$ and $\bar{\beta}_{p+1}$ are set to zero. Our observations match the assumption behind the *INTERP* strategy (see fig. 4.4).

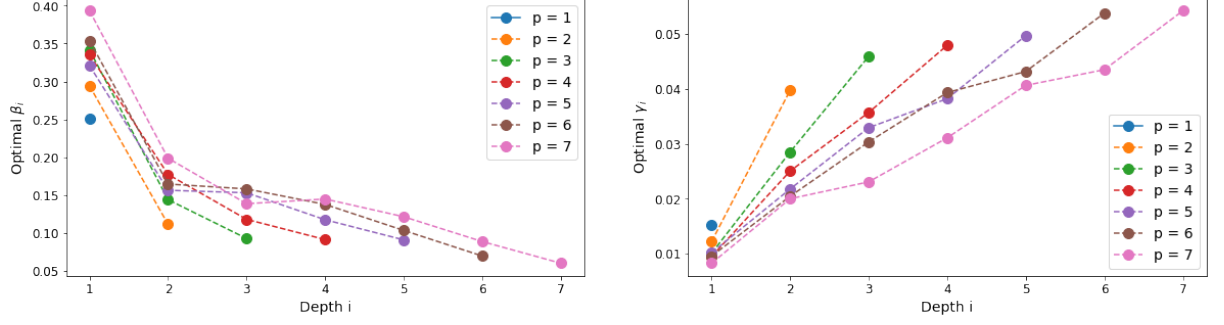


Figure 4.4: Optimal QAOA parameters at different depth p for an instance with $n = 8$ charges. We can observe that the shape of parameter curve at some depth doesn't significantly change when the depth is incremented by 1.

Optimization method

For the parameter optimization we compared different local optimization methods implemented in the *scikit* package for Python (see fig. 4.5). Methods were evaluated on two metrics. The first one is *optimization cost*. We measured the cost in *number of function evaluations* made by the method until convergence. The second metric is the *quality* of the returned optimum that is measured by the approximation ratio:

$$\alpha = \frac{L(\beta^o, \gamma^o)}{c^*}, \quad L((\beta), \gamma) = \langle (\beta), \gamma | C | (\beta), \gamma \rangle \quad (4.23)$$

where c^* is the optimum cut and β^o, γ^o are parameter values returned by the evaluated local routine. In our experiment the exact solutions c^* were obtained with the *dynamic program* for the scheduling problem [Sahni, 1976].

According to our simulations, *Nelder-Mead* is the best choice both in terms of the solution quality and the expected runtime. However, we didn't evaluate its resilience to the imprecision in the evaluation of the loss $L(\beta, \gamma)$.

In the *simulations* of the QAOA on a perfect hardware the *stochastic noise* appears when the loss function is approximated by the average energy of samples returned by a quantum computer:

$$L(\beta, \gamma) = \frac{1}{M} \sum_{i=1}^M C(X_i), \quad X_i \sim \mathbb{P}(\beta, \gamma) \quad (4.24)$$

In *realistic settings* the noise is induced both by the hardware imperfections *and* by the stochastic imprecision. In the comparative study reported in [Wilson *et al.*, 2021] Nelder-Mead was shown to be particularly affected by the noise.

Taking in account previously mentioned observations, we suggest a protocol that consists of a manually tuned *initialization strategy* and a privileged optimization method. For $p = 1$ our initialization strategy recycles already known optimal parameters. For $p > 1$ we adopt the *INTERP* approach as its motivation seems to be valid in our application. We also suggest to use the Nelder-Mead local routine as it has good performance and low cost compared to other evaluated methods.

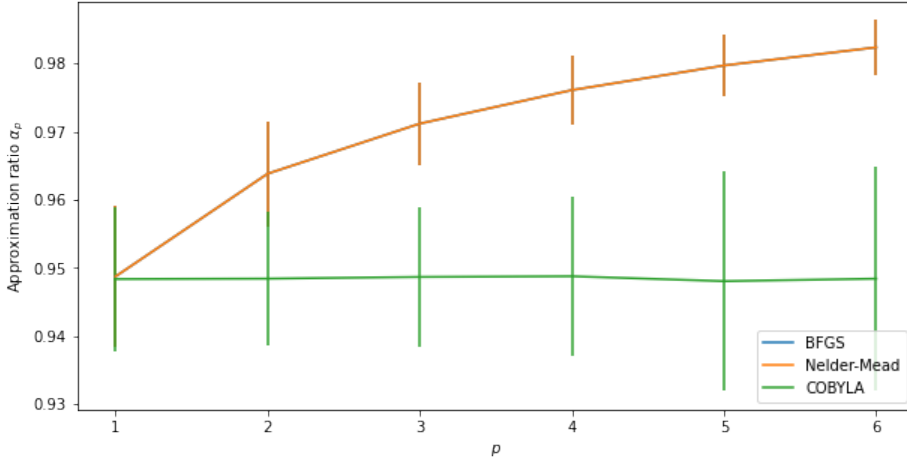
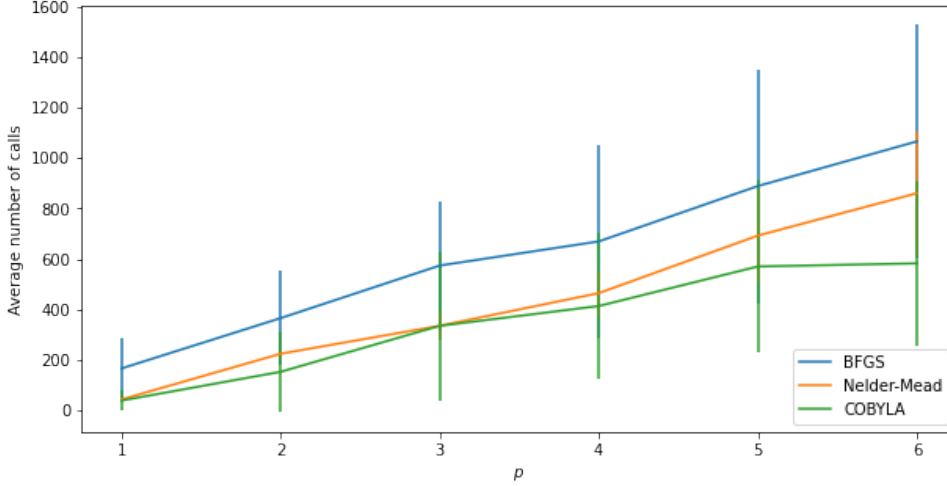
Average QAOA performance on graphs with $|V| = 10$ nodes for different numerical optimization methodsOptimization cost on graphs with $|V| = 10$ nodes for different numerical optimization methods

Figure 4.5: A comparison of local optimization routines for the training of QAOA parameters. Vertical lines correspond to the error bars. The derivative-free Nelder-Mead method [Gao and Han, 2012] achieves almost the same quality as the quasi-Newton BFGS method [Fletcher, 1987] (on the top figure Nelder-Mead curve hides the BFGS one). Both routines perform better than the frequently used COBYLA optimizer [Powell, 1994]. Still Nelder-Mead needs less time to converge.

4.4.3 Performance evaluation

As expected, we have observed that QAOA performance grows with the depth p (see fig. 4.6). The increase of the approximation ratio with p also indicates that our approach for parameter optimization finds fairly good parameter values. Indeed, as the parameter optimization gets more challenging with p a poor strategy can lead to the *decrease* in the quality of returned solutions [Shaydulin and Alexeev, 2019].

Surprisingly, we also noticed that the approximation ratio gets better with the size n of the scheduling instance. As illustrated on Figure 4.7 the similar behavior is observed for the random assignment algorithm. From the curves on Figure 4.7 we can conclude that QAOA outperforms

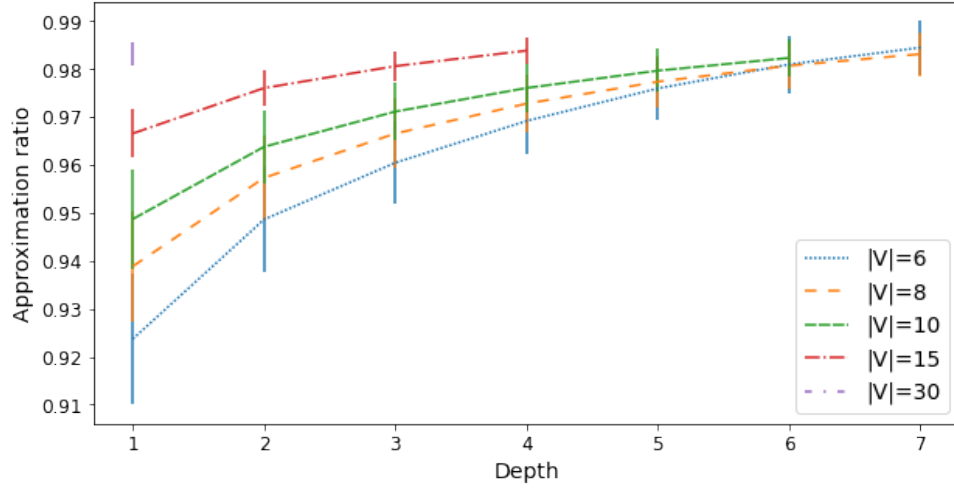


Figure 4.6: QAOA performance on instances of MaxCut generated from the Smart Scheduling on realistic data. The running time of the simulations scales exponentially with size on the instance.

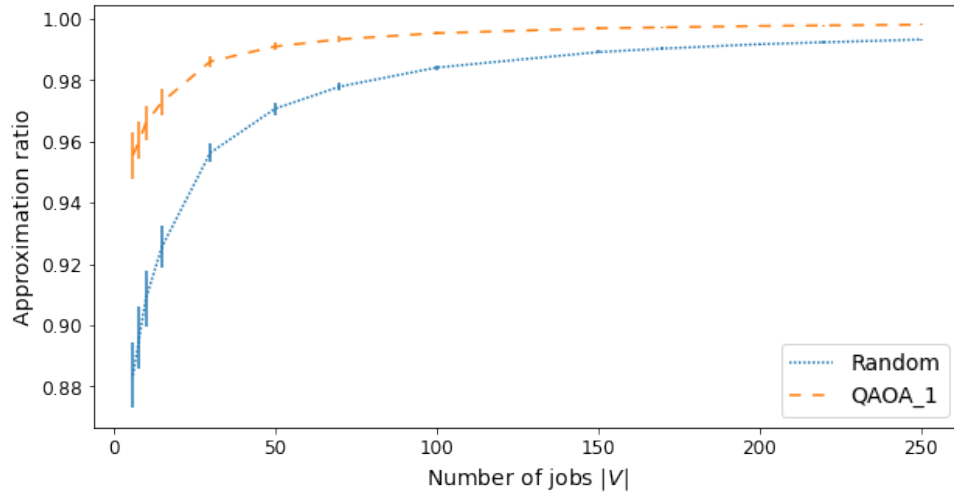


Figure 4.7: Approximation ratio achieved by QAOA₁ and a random algorithm on Smart Scheduling instances of different sizes n .

the randomized heuristics. Yet, the advantage vanishes with n . The good performance of the randomized algorithm suggests that the difficulty of the instances under consideration decreases with the size.

In what follows we sketch a probabilistic argument supporting this conclusion. This argument is *not a formal proof* but rather a *reflection* about the reasons that can explain the scaling behavior.

Why the problem gets easier with size?

We have an instance of MaxCut on complete weighted graph $G = (V, E)$. A complete graph on n nodes has $m = \frac{n(n-1)}{2}$ edges. A randomized algorithm on such instance returns the cut with

expected value

$$\nu_r = \frac{\sum_{(u,v) \in E} w_{u,v}}{2} \quad (4.25)$$

Let's assume that the weights of edges are independently sampled from some probability distribution $w_{u,v} \sim \mathcal{D}(\mu, \sigma^2)$ with mean value μ and variance σ^2 . We aim to evaluate the scaling of the average-case behavior of the randomized algorithm with n .

In a complete graph a partition can't cut more than $L = \frac{n}{2} \times \frac{n}{2} = \frac{m}{2} + \frac{n}{4}$ edges. The worst performance of the randomized algorithm corresponds to the case when the optimal cut has *exactly* L edges and, moreover, these are edges of the highest weights. In other words, we can bound the value of the optimal cut by:

$$\nu^* \leq \max_{\substack{E_l \subset E \\ |E_l|=L}} \sum_{e \in E_l} w_e \quad (4.26)$$

The number of subsets E_l that have the cardinality L can be evaluated with the binomial coefficient:

$$N = \binom{|E|}{L} = \binom{m}{m/2 + n/4} \leq \binom{m}{m/2} \leq \binom{2L}{L} \approx \frac{2^{2L}}{\sqrt{\pi m}} \quad (4.27)$$

The last approximate equality in (4.27) is derived from the *Stirling's formula* [Keller and Vanden-Broeck, 2007].

For each subset E_l we introduce new random variable

$$X_l = \frac{\sum_{e \in E_l} w_e}{L} \quad (4.28)$$

The central limit theorem states that for big L variables X_l are distributed according to the normal distribution [CLT, 2008]:

$$X_l \sim \mathcal{N}(\mu, \frac{\sigma^2}{L}) \quad (4.29)$$

The *cumulative distribution function (CDF)* $\Phi(x) = \mathbb{P}[X_l \leq x]$ for the distribution $\mathcal{N}(\mu, \frac{\sigma^2}{L})$ can be written as:

$$\Phi(x) = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{\sqrt{L}(x - \mu)}{\sigma\sqrt{2}} \right) \right] \quad (4.30)$$

where the error function $\operatorname{erf}(x)$ satisfies the bound [Chiani *et al.*, 2003]:

$$\operatorname{erf}(x) \geq 1 - e^{-x^2} \quad (4.31)$$

Using this bound we can get for any $a > 0$:

$$\Phi(\mu + a) = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{\sqrt{L}a}{\sigma\sqrt{2}} \right) \right] \quad (4.32)$$

$$\geq 1 - \frac{1}{2} e^{-La^2/2\sigma^2} \quad (4.33)$$

We now consider N **independent** variables $[X_1, \dots, X_N]$ identically distributed as (4.29). Actually, the initial variables $X_l = \frac{\sum_{e \in E_l} w_e}{L}$ are not independent as two sets E_l may have non-empty intersection. We will come back to the impact of the independence assumption later in this section.

We evaluate the probability that the maximum value of these variables doesn't exceed the mean value by a fixed shift a :

$$\mathbb{P} \left[\max_{l \in [1, \dots, N]} X_l \leq \mu + a \right] = \mathbb{P} \left[\bigcap_{l \in [1, \dots, N]} (X_l \leq \mu + a) \right] \quad (4.34)$$

$$= \prod_{l \in [1, \dots, N]} \mathbb{P} [X_l \leq \mu + a] = \Phi(\mu + a)^N \quad (4.35)$$

In this expression both N and $\Phi(\mu + a)$ have an exponential dependence on the number L . In order to evaluate how this probability scales with L we use (4.33, 4.27):

$$\log \Phi(\mu + a)^N = N \log \Phi(\mu + a) \geq \frac{2^{2L}}{\sqrt{\pi L}} \log \left(1 - \frac{1}{2} e^{-La^2/2\sigma^2} \right) = A(L) \quad (4.36)$$

where $L = \frac{n^2}{4}$.

Unfortunately, the bound 4.36 is too loose. For big n it only leads to a trivial bound $\mathbb{P} [\max_{l \in [1, \dots, N]} X_l \leq \mu + a] \geq 0$.

The poor result is caused by the large value $N = \binom{|E|}{m}$. In reality, there is only $K = \binom{n}{n/2}$ partitions that cut exactly L edges. Intuitively, for identically distributed edge weights the optimal partition on large instances should be one that splits nodes on two equal parts. Assuming that only one of these partitions can lead to an optimal value, the probability that the optimal cut is not too far from the mean value becomes:

$$\mathbb{P} \left[\max_{l \in [1, \dots, K]} X_l \leq \mu + a \right] = \Phi(\mu + a)^K \geq e^{B(n)} \quad (4.37)$$

$$B(n) = \frac{2^n}{\sqrt{\pi n/2}} \log \left(1 - \frac{1}{2} e^{-n^2 a^2 / 2\sigma^2} \right) \quad (4.38)$$

$$K = \binom{n}{n/2} \quad (4.39)$$

The graph for $B(n)$ is shown on fig. 4.8. According to this plot, the probability that the optimum is close to the mean value converges to 1 for large n .

Finally, we recall that we have assumed that the variables X_l are *independent* while this is not exactly true in our setup. Indeed, two variables $X_1 = \sum_{e \in E_1} w_e / L$ and $X_2 = \sum_{e \in E_2} w_e / L$ can depend on the same terms. In such case they are positively correlated. Indeed, we can decompose:

$$X_1 = A + B \quad (4.40)$$

$$X_2 = A + C \quad (4.41)$$

$$(4.42)$$

where $A = \sum_{e \in E_1 \cup E_2} w_e / L$, $B = \sum_{e \in E_1 \setminus E_2} w_e / L$ and $C = \sum_{e \in E_2 \setminus E_1} w_e / L$ are independent from each other. The covariance of X_1 and X_2 is:

$$E[X_1 X_2] - E[X_1]E[X_2] = E[(A + B)(A + C)] - E[A + B]E[A + C] = E[A^2] - E[A]^2 > 0 \quad (4.43)$$

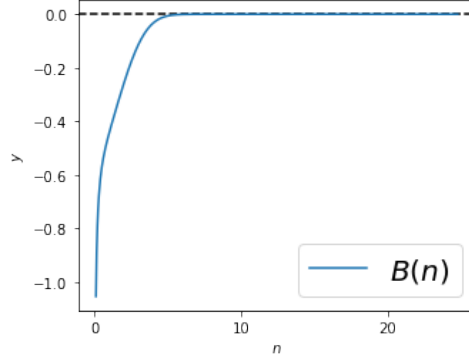


Figure 4.8: On both graphs we take $\sigma = 1$ and $a = \sqrt{2}$. The plot on the left corresponds to the bound $A(L) = \frac{2^{2L}}{\sqrt{\pi L}} \log(1 - \frac{1}{2}e^{-La^2/2\sigma^2})$.

On the right we show $B(n) = \frac{2^n}{\sqrt{\pi n/2}} \log(1 - \frac{1}{2}e^{-n^2 a^2/2\sigma^2})$

According to [Nadarajah and Kotz, 2008] the maximum of two Gaussian random variables with correlation r is distributed as:

$$p(x) = 2\phi(x)\Phi\left(\frac{1-r}{\sqrt{1-r^2}}x\right) \quad (4.44)$$

where $\phi(x)$ and $\Phi(x)$ are density and CDF for the standard normal distribution $\mathcal{N}(0, 1)$. Probability distributions for different positive correlation values r are shown on the figure 4.9. We observe that for positively correlated random variables the distribution weight "goes left", i.e. the expected maximum gets smaller. Motivated by this observation, we bound the maximum of *positively correlated* normal variables by the maximum of *independent variables* sampled from the same individual distributions.

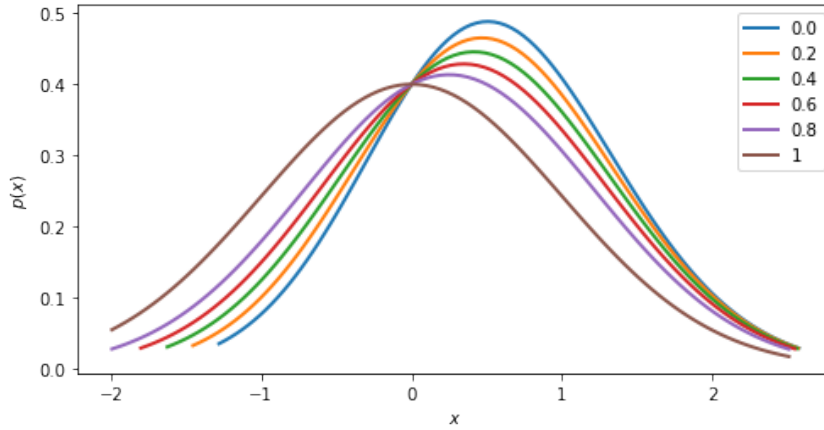


Figure 4.9: Distribution of $\max\{X_1, X_2\}$ for different correlation values $r = [0, \dots, 1]$. The value $r = 0$ corresponds to independent variables.

4.4.4 Possible improvements

Our QAOA protocol for the Smart Scheduling problem may be improved in several ways.

On one hand, we can profit from the concentration to design advanced training techniques to accelerate the the search of optimal parameters. For instance, a set of pretrained parameters for different instances of MaxCut [Shaydulin *et al.*, 2021] can be transfered to weighted instances [Shaydulin *et al.*, 2022]. We can use these results to gain in the *runtime* and *quality* of the optimization routine.

On the other hand, it is possible to enhance the performance of QAOA by adding some *classical preprocessing*. For instance, classical information about the optimum can be extracted from the solution of the SDP relaxation or, alternatively, the low-rank Burer-Monteiro relaxations. Warm-start approaches [Egger *et al.*, 2021, Tate *et al.*, 2020] suggest to integrate this classical information inside the initial state $|\psi_0\rangle$.

Chapter 5

Charging Task Selection

In the Smart Scheduling problem the starting time of the jobs J could be chosen freely. In some other popular scheduling problems jobs are provided with release dates $r_j, j \in J$ and deadlines $d_j, j \in J$. If the difference $d_j - r_j$ is strictly equal to the duration t_j of the job then it has to be executed *precisely* in the interval $[r_j, d_j]$. In such setup the decision-maker is given a set of jobs with intervals and it has to chose which jobs will be accepted for the execution.

5.1 Problem statement

In the Charge Task Selection problem each load request is given a predefined starting time $T_j^0 \in \mathbb{R}^+$ and processing time $t_j \in \mathbb{R}^+$. Charges are non-preemptive: once started they can't be interrupted until completion. If accepted, a job j brings a profit $p_j \in \mathbb{R}^+$. Decision-maker aims to select a *valid schedule* of highest profit. A schedule is valid if selected jobs are simultaneously satisfiable. For instance, if the charges are accomplished on a *single station* one can't accept two jobs i and j with overlapping time intervals $[T_i^0, T_i^0 + t_i] \cap [T_j^0, T_j^0 + t_j] \neq \emptyset$.

Charge Task Selection belongs to the class of *Interval Scheduling problems*. In essence, this class implements a demand-oriented philosophy: a set of mandatory constraints is given by clients and a logistic manager should satisfy them with minimum amount of resources. In the absence of additional constraints the Interval Scheduling problem on one station can be solved in polynomial time [Gavril, 1972]. However, most *modifications* are known to be NP-hard. The survey [Kolen *et al.*, 2007] describes different versions of Interval Scheduling problems and provides references to algorithmic solutions.

5.1.1 Group constraint

We consider here the Interval Scheduling Problem with additional *group constraints*. As before, each job comes with an interval $[T_j^0, T_j^0 + t_j]$. The goal is to select a maximal subset of tasks that can completed on a *single station*. Obviously, two demands with overlapping time intervals can't be simultaneously satisfied. In addition, each job is assigned to one *group*. A schedule is valid is it accepts *no more than one charge* from each group.

Although the one-per-group constraint seems rather artificial, it adapts to different realistic restrictions. For instance, such constraints model an environment where electric vehicles belong to different companies and no company should be privileged by the logistic manager.

Furthermore, a popular *Job Interval Selection* problem can be naturally formulated as Interval Scheduling problem with group constraint [Chuzhoy *et al.*, 2006]. In the smart charging

terminology the Job Interval Selection problem suggests for each vehicle a *list of intervals* when it can be charged. The decision-maker has to select at most one interval per vehicle while avoiding time conflicts.

5.1.2 Conflict graph representation

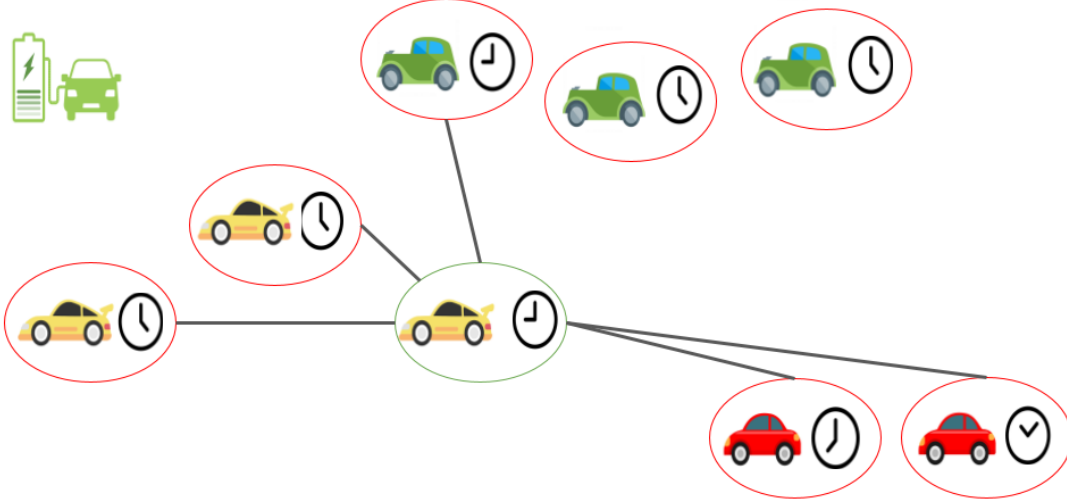


Figure 5.1: Example of the task selection problem. Colors of vehicles represent different groups. We depict the edges of the conflict graph for the yellow vehicle inside the green circle.

An instance of the Charge Task Selection problem with groups can be represented by a conflict graph $G = (V, E)$. Each node v_j corresponds to a job $j \in J$. There is an edge (v_i, v_j) between two nodes if the jobs i and j can't be simultaneously satisfied, i.e. they either belong to the same group or their intervals overlap (see fig. 5.1). The resulting graph is a union $G = (V, E_I \cup E_G)$ of an interval graph $G = (V, E_I)$ and disconnected cliques modeling the group constraint $G = (V, E_G)$. In literature the term *strip graphs* is used for the union of an interval graph and a graph made out of cliques [Halldórsson and Karlsson, 2006].

Each valid solution for the Charge Task Selection problem corresponds to an independent set on the conflict graph G . If we associate to each node a weight p_j task selection of maximum profit is equivalent to the maximum weighted independent set on G .

We denote by $k \in \mathbb{N}$ the cardinality of the largest group. For each instance one can create an equivalent one where each group has exactly k jobs. The transformation can be done by adding copies of some jobs. An optimum solution for the original instance can be extracted from the solution of the new instance. Therefore, from now we assume that each group contains exactly the same amount $k \in \mathbb{N}$ of jobs. We also take all profits equal $p_j = 1$. In other words, we want to maximize the *amount* of satisfied charges.

If each group has only one element the problem is polynomial. However, if $k \geq 2$ the problem is NP-Hard and it doesn't admit a polynomial time approximation scheme [Spieksma, 1999].

A greedy algorithm [Spieksma, 1999] achieves the approximation ratio of $1/2$. This ratio was improved to $\frac{e-1}{e}$ where $e = 2.718\dots$ is *Euler number* in [Chuzhoy et al., 2006].

5.2 Maximum Independent Set

An independent set (also called stable) in a graph $G = (V, E)$ is a subset of nodes $S \subset V$ such that none of them are connected by an edge: $\forall u, v \in S : (u, v) \notin E$. We denote by $I(G)$ the ensemble of all stable sets of the graph G .

A *Maximum Independent Set (MIS)* in $G = (V, E)$ is a stable S_{max} of largest cardinality:

$$S_{max} = \operatorname{argmax}_{s \in I(G)} |s| \quad (5.1)$$

The size $|S_{max}|$ is called the *stability number* of the graph and is conventionally denoted by $\alpha(G)$. *Maximum* independent sets have to be distinguished from *maximal* independent sets. A stable S is *maximal* if it can't be extended to a bigger independent set:

$$\forall v \in V \setminus S \quad \exists u \in S : (u, v) \in E \quad (5.2)$$

Obviously, a *maximum* independent set is *maximal* but the inverse is not true.

In the weighted version of MIS (referred as MWIS) a real number $w_v \in \mathbb{R}, v \in V$ is associated with each node. The goal is to chose an independent set of biggest weight:

$$S_{wmax} = \operatorname{argmax}_{s \in I(G)} |W(s)| \quad (5.3)$$

where for every set S the expression $W(S) = \sum_{v \in S} w_v$ is the total weight of all vertices in S .

Related problems

A *clique* C in a graph $G = (V, E)$ is a set of pairwise connected vertices: $\forall u, v \in C : (u, v) \in E$. The cardinality of a maximum clique in a graph G is called the *clique number* $\omega(G)$. Clearly, a maximum independent set on a graph $G = (V, E)$ is equivalent to a *maximum clique (MC)* in a *complement graph* $\overline{G} = (V, \overline{E})$ where $(u, v) \in \overline{E}$ if and only if $(u, v) \notin E$. Therefore, all results for one problem can be transposed for the second problem.

Graph coloring is another problem closely related to MIS and MC. This problem is an optimization over *proper vertex colorings*. A proper vertex coloring is a partition of the set of nodes V on $k \in \mathbb{N}$ independent sets $V = S_1 \sqcup \dots \sqcup S_k, S_i \in I(G)$. A minimal natural number k that allows a proper coloring is called chromatic number $\chi(G)$.

The sandwich theorem [Lovász, 1979] relates different graph characteristics mentioned above:

$$\omega(G) < \theta(\overline{G}) \leq \chi(G) \quad (5.4)$$

In this expression $\theta(\cdot)$ is the *Lovasz number* of the graph. A Lovasz number is equal to the value of some specific SDP program. Contrary to $\omega(G)$ and $\chi(G)$ that are NP-hard to compute, the Lovatz number can be evaluated in polynomial time.

5.2.1 Binary formulations of the M(W)IS problem

A subset of vertices S set can be represented by a binary characteristic vector $x^S \in \{0, 1\}^{|V|}$ such that:

$$x_v^s = \begin{cases} 1, & v \in S \\ 0, & v \in V \setminus S \end{cases} \quad (5.5)$$

M(W)IS can be formulated as *integer linear program* [Bomze *et al.*, 1999]:

$$\max \sum_{v \in V} w_v x_v \quad (5.6)$$

$$x_u + x_v \leq 1, \quad \forall (u, v) \in E \quad (5.7)$$

$$x_v \in \{0, 1\}, \quad \forall v \in V \quad (5.8)$$

More importantly for quantum algorithms, the MIS problem is equivalent to the following *QUBO*:

$$\min \quad x^t A x \quad (5.9)$$

$$x \in \{0, 1\}^{|V|} \quad (5.10)$$

where $A = A_G - I$ and A_G is the adjacency matrix of the graph [Bomze *et al.*, 1999]. For the weighted version the matrix A is:

$$a_{i,j} = \begin{cases} -w_i, & \text{if } i = j \\ -\frac{1}{2}(w_i + w_j), & \text{if } (i, j) \in E \\ 0, & \text{if } (i, j) \in \overline{E} \end{cases} \quad (5.11)$$

Motzkin and Strauss presented a similar continuous quadratic optimization problem [Motzkin and Straus, 1965]:

$$\max x^T A_G x \quad (5.12)$$

$$\sum_{v \in V} x_v = 1 \quad (5.13)$$

$$x_v \geq 0, \quad \forall v \in V \quad (5.14)$$

Interestingly, the optimum value of the *Motzkin Strauss problem* is precisely $1 - 1/\omega(G)$.

5.2.2 Different graph types

An instance of the Charging task Selection problem without group constraint corresponds to an *interval graph*. A graph is called an *interval graph* if there exists a map $I : V \rightarrow \mathbb{R}^+ \times \mathbb{R}^+$ that assigns to each node v an interval $[T_v^0, T_v^f]$ on the time line such that two nodes are connected by an edge if and only if the corresponding intervals overlap. Determining if $G = (V, E)$ is an interval graph can be done in linear time $O(|V| + |E|)$ [Habib *et al.*, 2000].

Interval graphs are a special case of more general *chordal graph*. In chordal graphs (also called triangulated graphs) any cycle $\{u_i\}_{i \in C}$ of length greater than four has a *chord*. *Chord* in a cycle is an internal edge (u_i, u_j) that connects two non-subsequent nodes: $j \neq i + 1$.

Chordal graphs in turn form a subset of *perfect graphs*. A graph is perfect if its clique number equals its characteristic number: $\omega(G) = \chi(G)$. Interestingly, the complement of a perfect graph is also perfect graph [Grötschel *et al.*, 1984].

When the group constraint is imposed, the Charging Task Selection problem corresponds to a *strip graph* - a union of interval graph $G = (V, E_I)$ and a graph made out of several cliques $G = (V, E_G)$.

Our real-world application correspond to a *strip graph*. On the other hand, some quantum hardware (such as neutral atom chip developed by Pasqal [Henriet *et al.*, 2020]) is particularly well suited to solve the MIS problem on a distinct family of graphs called *unit disks*. A graph is called *unit disk* if there exists a transformation mapping each node v to a point $p_v = (x_v, y_v)$ in two-dimensional Euclidian space such that two nodes u and v are connected by an edge: $(u, v) \in E$ if and only if corresponding points p_u and p_v are at distance at most one in \mathbb{R}^2 .

5.2.3 Complexity and approximations

The Maximum Independent Set problem on general graph is NP-complete [Karp, 1972]. Hence, known exact algorithms solving it have an exponential runtime in the worst case.

Exact algorithms

Most of exact algorithms are based on a *Branch & Bound* technique that uses bounds to explore the solution space more efficiently. For instance, the algorithm presented in [Balas and Yu, 1986] uses an MIS on induced triangulated subgraph to find a good lower bound. This technique was extended to weighted MIS in [Balas and Xue, 1991] and tested on instances with up to 2000 nodes.

The runtime performance of exact algorithms for MIS significantly varies depending on considered instances. Thus, *CLIQUEUR* solver developed in [Östergård, 1999] is fast on graphs with high density, while on sparse graphs it is outperformed by the Branch & Bound approach presented in [Held *et al.*, 2012].

Approximation

The approximation hardness of MIS was proven in several steps. Firstly, the work [Garey and Johnson, 1976] demonstrated that if one can approximate MIS to an arbitrary fixed ratio α then the problem admits a PTAS. As this was considered to be unlikely, MIS was not expected to have an approximation algorithm with constant α . A rigorous inapproximability result appeared in [Håstad, 1999, Zuckerman, 2007]. According to this result for all $\epsilon > 0$ there is no polynomial algorithm returning for every instance a stable S^* such that $|S_{max}|/|S^*| \leq n^{1-\epsilon}$ unless $P = NP$. The best known approximation algorithm [Boppana, 1990] achieves the ratio $|S_{max}|/|S^*| = O(n/\log^2 n)$ that asymptotically matches the inapproximability bound.

Special cases

The hardness and the inapproximability results mentioned above describe the *worst-case* complexity. As expected, they don't hold for various special families of the instances. For instance, there exist polynomial algorithms for MIS on *chordal graphs* [Gavril, 1972] and, more generally, for *perfect graphs* [Grötschel *et al.*, 1984].

On *strip graphs* MIS problem is *fixed-parameter tractable (FPT)* with respect to the number of groups k . In other words, for fixed k there exists an algorithm polynomial in $|V|$ and $|E|$ that solves the problem. One such algorithm based on dynamic programming was presented in [Halldórsson and Karlsson, 2006]. It has a runtime complexity $O(2^k n)$. Better time complexity is obtainable for the construction presented in [Bevern *et al.*, 2014].

For MIS on *unit disk* graphs a polynomial time approximation scheme based on local search was presented in [Chan and Har-Peled, 2012]. A iterative procedure [Nieberg *et al.*, 2004] implies a PTAS for the weighted case.

Finally, for random graphs sampled from the same family the numbers $\alpha(G)$ and $\omega(G)$ tend to concentrate. Indeed, it was shown in [Matula, 1976] that for Erdos-Renyi random graphs with density δ the clique number belongs to an interval

$$\lfloor M(n, \delta) \rfloor \leq \omega(G) \leq \lceil M(n, \delta) \rceil \quad (5.15)$$

$$(5.16)$$

with probability going to 1 where

$$M(n, \delta) = 2 \log_{\frac{1}{\delta}} n - 2 \log_{\frac{1}{\delta}} \log_{\frac{1}{\delta}} n + 2 \log_{\frac{1}{\delta}} \frac{e}{2} + 1 \quad (5.17)$$

5.2.4 Classical Heuristics

Maximum Independent Set is a very challenging computational problem, so even the most rapid *exact algorithms* can't tackle large instances. The *approximation algorithm* [Boppana, 1990] has poor guarantees and is relatively slow. As a consequence, in practical applications the MIS problem is usually solved with *heuristics*.

A recently-reported sophisticated heuristic based on binary search finds optimal solutions on real-world instances with about 10^6 nodes in less than a minute [Lu *et al.*, 2017]. It can tackle instances with up to 10^8 vertices in several days.

DIMACS benchmark

Heuristics are commonly characterized by experimental performance. A diversified benchmark for *Maximum Clique (MC)* solvers was suggested in the DIMACS challenge [Johnson and Trick, 1996]. Since then it became a reference test dataset in most works on algorithms for MIS and MC.

DIMACS benchmark contains sixty-five instances from eleven different families of graphs issued from wide variety of applications. **CFat** instances are issued from fault diagnosis, **Johnson** and **Hamming** from coding theory. **San** and **Sanr** are generated random graphs with predefined clique size. Another family of random graphs is **PHat**. **PHat** graphs have wider node degree spread and larger $\omega(G)$ than standard Erdos-Renyi graphs. **Brock** instances have maximum clique that is much larger than the expected size. **MANN** instances comes from covering problems. **Keller** graphs are based on Keller's conjecture on tiling hypercubes.

Greedy heuristics

Some heuristics such as [Lu *et al.*, 2017] are inspired by Branch & Bound with limited time budget. Another popular research direction is *greedy algorithms*. In a nutshell, greedy algorithms maintain some partial solution and iteratively add or remove nodes from it depending on the values of indicators.

A naive greedy heuristic [Johnson, 1974] suggests to add a new node to an independent set in order defined by degrees. For weighted case the popular greedy strategies consider nodes $v \in A$ according to one of the following orders [Held *et al.*, 2012]:

- **maximum weight:** $v = \operatorname{argmax}_{v \in A} w_v$
- **maximum static surplus:** $v = \operatorname{argmax}_{v \in A} \left[w_v - \sum_{u \in V, (u,v) \in E} w_u \right]$
- **maximum dynamic surplus:** $v = \operatorname{argmax}_{v \in A} \left[w_v - \sum_{\substack{u \in A \\ (u,v) \in E}} w_u \right]$

In these expressions A denotes the set of *available vertices*. If in a current step the partial solution is an independent set S^t then the set A is $V \setminus (S^t \cup \{u \mid \exists v \in S^t : (u, v) \in E\})$.

Greedy approaches are particularly fast compared to other heuristics. Moreover, an integration of sophisticated post-processing allows to achieve very good performances. For instance, a *Greedy Randomized Adaptive Search (GRAS)* heuristic combines local search with a greedy initialization [Feo *et al.*, 1994]. In experimental evaluation on random graphs with variable densities GRAS was able to find optimal or expected optimal solutions on most instances.

Local search

Greedy algorithms iteratively construct the solution. A distinct strategy is adopted by *local search* methods. A typical local search procedure is initialized by some feasible solution x^t . Thereupon it explores the local neighborhood $N(x^t)$ of the initialization point x^t . If a better solution $x^* \in N(x^t)$ is found, the methods makes a move towards it: $x^{t+1} = x^*$. Otherwise, the exploration either stops or moves to some suboptimal solution $x^{t+1} = \hat{x}$ with probability $p(x^t, \hat{x})$.

There exists a wide variety of local search methods with different notion of neighborhood $N(\cdot)$ and diverse modification strategies $p(x^t, \hat{x})$. For instance, *2-interchange heuristic* replaces a single vertex in a maximal stable x^t with two others if this improves the value of the solution. The work [Andrade *et al.*, 2008] presents a method that checks if there is a 2-improvement in a linear time. It also shows that such local search strategy is systematically able to improve a naive greedy solution.

Simulated Annealing is another way to apprehend the local search framework. In this meta-heuristic the probability of transition to a suboptimal solution $p(x^t, \hat{x})$ is proportional to the global temperature parameter T . Emulating the controlled cooling in metallurgy, the temperature T is slowly decreased. Simulated Annealing as well as greedy algorithms showed good performance for the *Maximum Clique* problem on dense graphs [Homer and Peinado, 1994].

Tabu search is a local strategy that use the short-term memory to restrict the set of *legal neighbors*. The basic version is sometimes tuned with *intensification* and *diversification* strategies that allows to explore the solution space more efficiently. A refined tabu search [Battiti and Protasi, 2001] demonstrated excellent results on the DIMACS benchmark: on most instances it found already known best solutions. Moreover, for some graphs the algorithm improved previous bounds.

Greedy and local search strategies and their various combinations usually achieve satisfactory performance on MIS problem. Up to know, global optimization methods such as *genetic algorithms* [Marchiori, 1998], continuous-based heuristics [Gibbons *et al.*, 1993] and *neural networks* [Grossi *et al.*, 2006] were unable to significantly outperform them. Nevertheless, a genetic algorithm presented in [Marchiori, 1998] showed results comparable to *simulated annealing* and *tabu search* on DIMACS benchmark. The continuous based heuristic [Gibbons *et al.*, 1993], in turn, doesn't require a parameter calibration that has a significant impact on performance of metaheuristics.

5.3 Quantum heuristics for Charging Task Selection

5.3.1 Encoding

The Maximum Independent Set problem is equivalent to the QUBO formulation (5.9). This formulation can be equivalently written as :

$$\max \sum_{v \in V} x_v - \lambda \sum_{(u,v) \in E} x_u x_v \quad (5.18)$$

$$x \in \{0, 1\}^{|V|} \quad (5.19)$$

where $\lambda = 2$.

Penalization

The value $\lambda = 2$ is sufficient to make solutions that are not independent sets $x^S, S \notin I(G)$ suboptimal. However, quantum heuristics don't return an *exact optimum* but rather some low-energy state. In this settings it is extremely important to properly discriminate unfeasible solutions in the objective function. Therefore, the penalty value λ is typically set to a larger value $\lambda \gg 2$. Yet too large value of λ will "flatten" the relative energy landscape around feasible solutions reducing the performance of heuristics. *Bisection search* can be used to accurately fix the value of the penalty term [Stollenwerk *et al.*, 2018]. The bisection search simply checks the feasibility of solutions returned by a quantum heuristic on formulations with different values λ .

Another approach to ensure feasibility was adopted in [Farhi *et al.*, 2020]. In this work the value λ is taken to be $\lambda = 1$ and independent sets are extracted from the output of QAOA by a post-processing procedure.

Modified mixer Hamiltonian

Furthermore, the feasibility of solution returned by a quantum heuristic can be achieved by an appropriate modification of the driver Hamiltonian. The original QAOA paper [Farhi *et al.*, 2014a] suggests to use the mixer $\bar{U}_M(\beta) = e^{i\beta B'}$ where

$$\langle x|B'|x'\rangle = \begin{cases} 1, & x, x' \in I(G), \quad x \text{ and } x' \text{ differ in one bit} \\ 0, & \text{otherwise} \end{cases} \quad (5.20)$$

The operator $\bar{U}_M(\beta)$ preserves the evolution in the feasible subspace. However, it is not clear how to design a circuit implementing such evolution. Taking in account the implementability, the work [Hadfield, 2018] introduces a sequential mixing $U_M^s(\beta) = \prod_{v \in V} U_{M,v}^s(\beta)$. For each node with l neighbors $N(v)$ the unitary $U_{M,v}^s(\beta)$ is $e^{i\beta B_v}$ where the operator B_v is:

$$B_v = \frac{1}{2^l} X_v \prod_{u \in N(v)} (I + Z_u) \quad (5.21)$$

5.3.2 Maximum Independent set on neutral atoms

For *unit disk* instances the MIS problem (UD-MIS) is *native* for *neutral atom platforms* [Henriet *et al.*, 2020, Pichler *et al.*, 2018]. Using laser control system such platform creates a energy landscape corresponding to the Hamiltonian:

$$H = \sum_{i=1}^N \frac{\hbar\Omega}{2} X_i - \sum_{i=1}^N \frac{\hbar\delta}{2} Z_i + \sum_{j < i} \frac{C_6}{|\mathbf{r}_i - \mathbf{r}_j|^6} \frac{Z_i + 1}{2} \frac{Z_j + 1}{2}. \quad (5.22)$$

where r_i is a coordinate of a node in the unit disk embedding, Ω is Rabi frequency and δ is detuning of the laser system. The energy term for pair interactions prevents two atoms to be in an excited state if they are at distance at most $r_b = (C_6/\hbar\Omega)^{\frac{1}{6}}$. At the same time, the interaction is negligible for other pair. This phenomenon is called *Rydberg blockage* [Urban *et al.*, 2009]. The Rydberg blockade imposes the system to be in a state that corresponds to an independent set of the chosen unit disk layout.

A neural atom processor works in an analog mode. Using laser control, one can drive the system to an approximate solution of the UD-MIS problem.

Embedding in unit disk

In order to apply this technique to generic Charge Task Selection instances one has to find a proper *embedding* of the instance into a unit disk. In other words, we have to find a mapping (usually referred as *realization*) from the set of nodes to a set of points in 2D. If we denote by $d(p_1, p_2)$ a Euclidean distance between two-dimensional point $p_1 \in \mathbb{R}^2$ and $p_2 \in \mathbb{R}^2$ the *embedding problem (EP)* can be formulated as:

$$\text{Find } \{(\mathbf{v}_i)\}_{i \in V} \quad (5.23)$$

s.t.

$$\rho \leq d(\mathbf{v}_i, \mathbf{v}_j) \leq r_b, \quad \forall (i, j) \in \mathbb{E} \quad (5.24)$$

$$d(\mathbf{v}_i, \mathbf{v}_j) > r_b, \quad \forall (i, j) \notin \mathbb{E} \quad (5.25)$$

$$\mathbf{v}_i \in [0, \bar{L}]^2, \quad \forall i \in V \quad (5.26)$$

where $\rho \sim r_b/3$ is the minimal spacing between separated atoms and \bar{L} is a maximum square area to place atoms in a quantum chip. In a physical system typical values of the parameters are $r_b \approx 15\mu m$ for the Rydberg blockade radius, $\rho \approx 5\mu m$ and $\bar{L} \approx 100\mu m$ for the platform characteristics.

We remark that the embedding problem doesn't always admit a feasible solution. For instance, a limited number of vertices can be placed in a disk $D(\rho, r_b)$. Therefore, a node with a sufficiently high degree can't be embedded in such layout.

Besides, in order to leave a room for quantum advantage embedding should be *efficiently computable*.

Because of the constraints (5.24, 5.25) the formulation (EP) is non-convex, thus it is expected to be difficult [Park and Boyd, 2017]. Indeed, it is NP-hard to determine if the problem admits a solution at all [Breu and Kirkpatrick, 1998]. Still, in practice (EP) can be transformed to a *mixed integer linear program* and optimized by a conventional solver such as CPLEX [Cplex, 2009]. Unfortunately, our experiments revealed that the solution time of this (EP) formulation is too large so it can compromise the potential quantum advantage.

In [Dalyac *et al.*, 2021] we suggested a more restrictive formulation (EPR):

$$\text{Find } \{(x_i, y_i)\}_{i \in V} \quad (5.27)$$

s.t.

$$\bigvee_{\phi \in \Phi} f_\phi(x_i, x_j, y_i, y_j), \quad \forall (i, j) \in E \quad (5.28)$$

$$(|x_i - x_j| > r_b) \vee (|y_i - y_j| > r_b), \quad \forall (i, j) \in \bar{E} \quad (5.29)$$

$$x_i, y_i \in [0, \bar{L}], \quad \forall i \in V \quad (5.30)$$

where $\Phi \subset [0, 2\pi]$ is a manually fixed discrete set of angles and the function $f_\phi : \mathbb{R}^4 \rightarrow \mathbb{B}$ indicates that the point (x_j, y_j) is on the radius ϕ of the disk $D(\rho, r_b)$ centered in (x_i, y_i) :

$$f_\phi(x_i, x_j, y_i, y_j) = (\rho \cos(\phi) \leq |x_i - x_j| \leq r_b \cos(\phi)) \wedge (\rho \sin(\phi) \leq |y_i - y_j| \leq r_b \sin(\phi)) \quad (5.31)$$

Constraints in the new formulation (EPR) are stronger than the ones in the original (ER). Therefore, the program (EPR) can be infeasible even if the graph actually admits a valid unit disk realization.

On the other hand, (EPR) contains only "or" and "and" logic combinators and absolute values of linear terms. It makes the formulation more tractable by the CPLEX solver. We tested

the modified embedding strategy on 100 instances of the Charge Task Selection problem with number of nodes from 12 to 15. CPLEX was able to solve the (EPR) problem in 60s for 84 instances. We remark that our test set contained only small graphs. For instances of realistic size the embedding procedure needs further refinements to allow a competitive runtime behavior.

Once an instance is properly embedded in the unit disk layout, the MIS problem can be directly solved on a neutral atom machine. A recent work [Dalyac and Henriët, 2022] reported a procedure that finds in polynomial time for graphs with maximum degree 6 an embedding in unit disk of *three dimensions* where Rydberg atoms are placed in 3D atomic arrays.

If an instance has no UD realization it can be addressed by some *classical-quantum heuristic* that outsources a part of computation to a quantum chip. One can imagine an heuristic implementing a *subgraph approach* similar to one in [Balas and Yu, 1986] where Branch & Bound procedure uses independent sets on induced triangulated subgraphs. A classical-quantum heuristic for general MIS can extract unit disk subgraphs from an instance and call a neutral atom chip to approximate MIS on the subgraph.

5.3.3 Numerical results

Experimental setup

In order to evaluate the performance of quantum heuristics on the MIS problem we created a benchmark that contains random regular graphs with degrees in $[3, 5, 7, 9]$ and sizes ranging from 100 to 1000. We also included DIMACS graphs and real-world instances issued from a simulator of the charging demand scenarios developed internally by EDF.

A full scale simulation of QAOA and Quantum Annealing is limited to small instances ($n \leq 30$). Therefore, we decided to evaluate the performance of RQAOA. Contrary to QAOA, RQAOA can be efficiently simulated at depth $p = 1$ (see section 3.6). Due to the efficient simulation, RQAOA at depth $p = 1$ is a purely classical algorithm. Nevertheless, our analysis sheds a light on the typical behavior of quantum heuristics.

Our implementation of RQAOA₁ is in a sense the simplest possible one. It uses the analytic formula (3.37) to evaluate the mean values $\langle Z_u \rangle$ and $\langle Z_u Z_v \rangle$. In the first iteration of the recursive variable elimination QAOA parameters are optimized with a global *differential evolution* method. For the following steps the *local Nelder-Mead* routine is used. Conducting numerical experiments we observed that the parameter optimization step is a genuine bottleneck for our implementation. Indeed, without any specific knowledge about the neighborhood of the optimal solution a global method takes a long time to find a sufficiently good initial point. Local Nelder-Mead optimization requires less function evaluations to converge. However, for each n^2 pairs the values $\langle Z_u Z_v \rangle$ are computed in $O(D_{max})$ with the formula (3.37). So for dense graphs $d = \Omega(n)$ the runtime becomes prohibitive for graphs with more than 100 nodes.

We believe that more refined parameter optimization strategies (such as described in the section 3.5.7) can accelerate the program and increase the performance of RQAOA.

RQAOA performance

In the beginning we compared RQAOA₁ to the *approximation algorithm (CA)* from [Boppana, 1990] on 121 random regular graphs with up to 1000 vertices. For the approximation algorithm we used the implementation from the *networkx* Python package. We observed that on all but 7 instances (mostly of small sizes 100 and 300) RQAOA₁ returns a solution that is at least as

good as the one found by the classical routine. Moreover, on 108 instances RQAOA₁ is strictly superior.

The DIMACS benchmark is designed for the *Maximum Clique* problem. A program for MIS can solve the Maximum Clique problem on $G = (V, E)$ by returning an independent set in a complement graph $G = (V, \overline{E})$. The complexity of our program for MIS significantly grows with the number of edges in the graph. For this reasons, we were able to obtain numerical results only for 40 out of 65 DIMACS instances that are either small or sufficiently sparse.

The experiment on DIMACS graphs revealed that RQAOA at $p = 1$ shows good performance on some particular instances (such as ones from the **Hamming** family). Still it returns a poor approximate solution in many cases. This implies that it is necessary to increase the depth p in order to observe the quantum advantage.

On the Charge Task Selection problem we compared RQAOA to an exact algorithm as well as to the classical heuristic both presented in [Held *et al.*, 2012] (see fig. 5.2). The exact algorithm is based on Branch & Bound, it can tackle instances with up to 500 nodes. The reference heuristic combines a *greedy initialization* with *2-interchange* local search. In order to make a fair evaluation, we decided to improve the RQAOA solutions with the same local search routine. Our results demonstrate that RQAOA, while used as initialization for the local search routine, can improve the solution compared to the classical method.

In summary, our numerical experiments about RQAOA applied to MIS provides encouraging results. While we were unable to definitely outperform classical heuristics, the obtained solutions were still competitive in many cases. Moreover, our analysis was restricted to RQAOA with depth $p = 1$. We believe that the performance of the algorithm can be significantly increased if we consider QAOA ansatzes with higher depth $p > 1$.

On the down side, we have observed that the solution quality is not robust across instances. Therefore, the selection of right usecases that lead to instances well-suited for quantum heuristics remains an extremely important challenge for the near-future quantum computing.

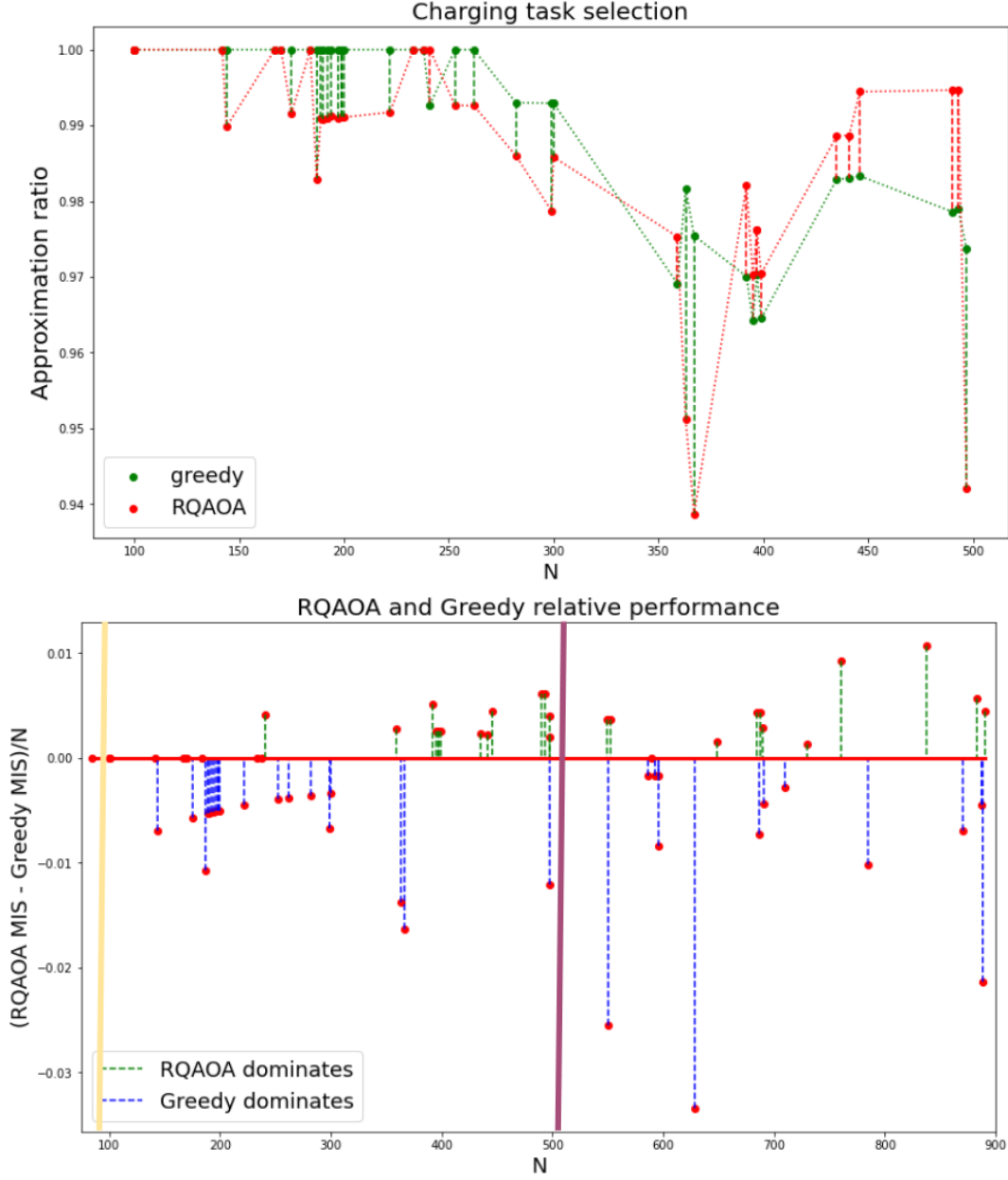


Figure 5.2: Comparison of RQAOA with a classical heuristic on Charge Task Selection problem. On the top panel is shown the exact approximation ratio obtained by both algorithms. The bottom panel illustrates the relative performance of two heuristics in the regime $|V| > 500$ marked by a violet line. In this regime the exact algorithm gets too slow. Vertical yellow line shows the limit on size of instances that can be analyzed by a full-scale simulation of quantum evolution. The full-scale simulation is necessary to get the output of such heuristic as QAOA.

Instance	$ V $	$ E $	exact $\omega(G)$	RQAOA
brock200_1	200	14834	21	17
brock200_2	200	9876	12	9
brock200_3	200	12048	15	10
brock200_4	200	13089	17	13
brock400_1	400	59723	27	17
brock400_2	400	59786	29	18
brock400_3	400	59681	31	17
brock400_4	400	59765	33	16
c-fat200-1	200	1534	12	12
c-fat200-2	200	3235	24	23
c-fat200-5	200	8473	58	F
hamming6-2	64	1824	32	32
hamming6-4	64	704	4	4
hamming8-2	256	31616	128	128
hamming8-4	256	20864	16	10
hamming10-2	1024	518656	512	510
johnson8-2-4	28	210	4	4
johnson8-4-4	70	1855	14	8
johnson16-2-4	120	5460	8	8
keller4	171	9435	11	8
MANN_a9	45	918	16	12
MANN_a27	378	70551	126	117
MANN_a45	1035	533155	345	330
p_hat300-1	300	10933	8	6
p_hat300-2	300	21928	25	19
p_hat300-3	300	33390	36	28
san200_0.7_1	200	13930	30	15
san200_0.7_2	200	13930	18	12
san200_0.9_1	200	17910	70	45
san200_0.9_2	200	17910	60	35
san200_0.9_3	200	17910	44	30
san400_0.5_1	400	39900	13	F
san400_0.7_1	400	55860	40	20
san400_0.7_2	400	55860	30	F
san400_0.7_3	400	55860	22	12
san400_0.9_1	400	71820	100	50
sanr200_0.7	200	13868	18	15
sanr200_0.9	200	17863		35
sanr400_0.5	400	39984	13	8
sanr400_0.7	400	55869		17

Table 5.1: Performance of RQAOA on DIMACS instances of the Maximum Clique problem. "F" entry for RQAOA signifies that the algorithm returned an infeasible solution on the corresponding instance. The feasibility issue can be addressed by increasing the penalty for infeasible solutions or with a special postprocessing.

Chapter 6

Hybrid Quantum-Classical Decomposition Scheme

According to current estimations, the near-future quantum hardware will have a relatively small number of qubits with limited connectivity [Preskill, 2018]. Besides, existing algorithms adapted to NISQ devices can deal with only several special types of applications. In the context of combinatorial optimization attainable applications are the problems that can be formulated as the optimization of an unconstrained quadratic binary function. Moreover, in order to execute the algorithm on a quantum hardware we have to specify the mapping between the connectivity pattern presented by quadratic terms and the hardware layout.

This obstructions hardly restrict the spectrum of problems that can be addressed directly by quantum algorithms. In order to circumvent these limits, for most applications quantum hardware is assisted by classical routines. Classical routines can be used in preprocessing, postprocessing as well as be directly integrated in *variational algorithms* such as QAOA [Farhi *et al.*, 2014a] and VQE [Peruzzo *et al.*, 2014]. An algorithm that combines quantum and classical parts is called *hybrid algorithm*.

Embedding

The famous example of a classical preprocessing is the *DWave embedding* procedure that maps logical variables to chains of physical qubits in the hardware layout [Choi, 2008]. Previously (see section 5.3.2), we presented a classical program that finds a *unit disk* representation required to address the problem on neutral atom platforms for instances of the Maximum Independent Set problem [Pichler *et al.*, 2018, Henriët *et al.*, 2020]. In general, embedding extends the applicability of quantum algorithms to more general classes of non-native instances. It is crucial for real-world applications.

In addition to the management of hardware constraints, hybridization can be used to address *more complicated formulations* than a vanilla QUBO. For instance, sophisticated hybrid approaches were suggested to handle *linear inequality constraints* [Braine *et al.*, 2021, Yonaga *et al.*, 2020].

Integration of inequality constraints

Management of inequality constraints that are often present in optimization problem is essential but also very challenging. Indeed, contrary to a linear equality constraint $a^T x + b = 0$ that can be straightforwardly integrated in the QUBO formalism with the penalty term $(a^T x + b)^2$,

penalization of inequalities requires additional *slack variables*. With slack variables an inequality constraint

$$a^T x + b \leq 0 \quad (6.1)$$

is transformed to

$$a^T x + b + s = 0 \quad (6.2)$$

$$s \geq 0 \quad (6.3)$$

The slack variable s is in general non-binary (and even non-integral). Usually, we can estimate only a poor upper bound $s \leq u \leq 2^k$. If the parameters a and b are integer-valued, the upper bound allows to represent s with a binary expansion $s = \sum_{q=0}^k 2^q y_q$, $y \in \{0, 1\}^k$. In a naive approach for the management of inequality constraint the penalty term $(a^T x + b + \sum_{q=0}^k 2^q y_q)^2$ is directly integrated in the objective function. Such approach requires additional $k = \lceil \log_2 u \rceil$ strongly-connected qubits. Thus, it dramatically reduces the size of the instances that can be tackled on the quantum hardware.

To deal with this issue, the work [Yonaga *et al.*, 2020] suggests to use a formulation with auxiliary *continuous variables* for slacks. The resulting objective function over binary *and* continuous variables is further optimized by a hybrid heuristic. This hybrid heuristic implements the *alternating direction method of multipliers* assisted by a quantum QUBO optimizer. In the work [Yonaga *et al.*, 2020] this method was illustrated on the *Quadratic Knapsack* application.

Motivated by the need for management of slack variables, the work [Braine *et al.*, 2021] presents a hybrid heuristic that handles *mixed binary problems (MBO)*:

$$\min_{\substack{x \in \{0,1\}^n \\ y \in \mathbb{R}^m}} x^t A(y) x + b(y)^T x + c(y) \quad (6.4)$$

where quadratic coefficients $A : \mathbb{R}^m \rightarrow \mathbb{R}^n \times \mathbb{R}^n$ and linear coefficients $b : \mathbb{R}^m \rightarrow \mathbb{R}^n$ depend on continuous variables.

The hybrid algorithm for MBO is a modified version of some variational quantum algorithm for QUBO (such as QAOA or VQE). For fixed values of $y \in \mathbb{R}^m$ quantum circuit is used to find good values for *binary* variables. The *continuous* variables $y \in \mathbb{R}^m$ are optimized in the same classical loop that optimizes the *parameters* of the quantum circuit (β and γ for QAOA). The heuristic iteratively modifies binary and continuous variables until the convergence condition is satisfied. In [Braine *et al.*, 2021] this hybrid method was applied to the *transaction settlement* problem.

A different approach presented in [Ohzeki, 2020] is designed to address fully-connected Ising models with the full connectivity that follows from the constraint penalization. This approach uses the *Hubbard-Stratanovich transformation* to replace the original quadratic terms by linear terms with stochastic coefficients. The values of coefficients are determined in an iterative procedure that evaluates the energy expectation value conditioned on the coefficient values by making calls to quantum annealer. In [Yu and Nabil, 2021] the resulting hybrid procedure was applied to the *bus charging scheduling* problem. It was shown that this approach allows to use the D-Wave quantum annealer to handle much bigger instances than if the naive penalization of constraints is applied.

Warm-starting quantum algorithms

Hybridization approaches presented above extend the *applicability* of quantum algorithms. Alternatively, the integration of classical routines can be used to enhance the *performance* of quantum heuristics. For instance, several *warm-start* techniques were proposed for QAOA on the Maximum Cut problem [Egger *et al.*, 2021, Tate *et al.*, 2020]. These techniques use specially-designed initial state $|\psi_0\rangle$ instead of the trivial uniform superposition. The preparation of the initial state is guided by the information computed classically. In [Egger *et al.*, 2021] two alternative options are presented. The first one involves the solution of the continuous relaxation of the quadratic program. The second approach employs the solution returned by the *Goemans-Williamson algorithm*. The *Goemans-Williamson* algorithm is a rounding of the *SDP relaxation*. Alternatively, *Burer-Monteiro* relaxation is used in the warm starting presented in [Tate *et al.*, 2020].

Finally, we can design hybrid methods where quantum routines are used to *accelerate* classical schemes. A version of quantum-assisted Branch & Price was presented in [Svensson *et al.*, 2021]. In this work quantum heuristics are used to approximately solve the *master problem*. In the next section we suggest a different *original way* to integrate quantum routines in the Branch & Price. We believe that our hybrid approach can improve the runtime of the routine compared to the fully classical procedure.

6.1 Integer Linear Programs

A huge variety of real-world optimization problems can be modeled as *integer linear programs*. An *Integer Linear Program (ILP)* in general form is an optimization problem that is written as follows:

$$(ILP) : \quad \min c^T x \quad (6.5)$$

$$Ax = b \quad (6.6)$$

$$Dx \leq d \quad (6.7)$$

$$x \in \mathbb{N}^n \quad (6.8)$$

where A and D are matrices with real values.

For instance, we can consider a delivery problem where a set of N items of weights ω_i , $i \in [1, \dots, N]$ have to be assigned to one of M trucks of different capacities C_t , $t \in [1, \dots, M]$. We use binary variables x_i^t that indicate that the item i is delivered by the truck t . Each truck, if used, generated the cost c_t . We denote by y_t the variable indicating that the truck t was selected for delivery. We wish to assure the delivery demands at lowest cost:

$$\min c^T \mathbf{y} \quad (6.9)$$

$$\sum_{t=1}^M x_i^t = 1, \quad i \in [1, \dots, N] \quad (6.10)$$

$$\sum_{i=1}^N \omega_i x_i^t \leq C_t y_t, \quad t \in [1, \dots, M] \quad (6.11)$$

$$x_i^t \in \{0, 1\}, y_t \in \{0, 1\} \quad i \in [1, \dots, N], t \in [1, \dots, M] \quad (6.12)$$

One conventional way to exactly solve an integer linear program is *Branch & Bound* [Burke and Kendall, 2002]. At each iteration the Branch & Bound algorithm considers different restrictions of the feasible

subspace. In order to evaluate if a particular subspace S is *promising*, i.e. if it can contain the optimal solution, the procedure uses *bounds*. Bounds indicates the range $[f_{min}, f_{max}] \subset \mathbb{R}$ of the values that the linear function $f(x) = c^T x$ take on the elements of the subspace S . For the procedure to be practical, bounds have to be efficiently computable.

6.1.1 Linear relaxation

In most implementations bounds are obtained via the *linear relaxation (LP)* of the integer linear program:

$$(LP) : \quad \min c^T x \quad (6.13)$$

$$Ax = b \quad (6.14)$$

$$Dx \leq c \quad (6.15)$$

$$x_i \geq 0, \quad x_i \in \mathbb{R} \quad (6.16)$$

In the linear relaxation we optimize the objective function over continuous variables instead of integer ones. Contrary to the problem over integer variables that is NP-hard, *the continuous relaxation can be efficiently solved*. In practice the *simplex method* (even if exponential in the worst case) is particularly efficient [Nash, 2000].

Obviously, the optimum value ν_{LP} of the linear relaxation is smaller or equal to the optimum ν_{ILP} of (ILP). The difference $g = \nu_{ILP} - \nu_{LP}$ is called the *relaxation gap*. The solution of the (LP) provides a lower bound for ν_{ILP} . The tightness of the lower bound is characterized by the relaxation gap.

Linear relaxation is not the only efficient way to evaluate the bounds of (ILP). Furthermore, in many cases the linear relaxation provides very poor bounds. For instance, if we consider an (ILP) for the Maximum Independent Set problem on a complete graph K_n :

$$\max \sum_{i \in V} x_i \quad (6.17)$$

$$x_i + x_j \leq 1, \quad \forall i \in V, j \in V, i \neq j \quad (6.18)$$

$$x_i \in \{0, 1\}, \quad \forall i \in V \quad (6.19)$$

the linear relaxation has the value $n/2$ (achieved in the point $x = [0.5, \dots, 0.5]$) while the optimum of the integer problem is 1.

Better bounds may be obtained via the *Lagrangian approach* where only a part of the constraints (coupling constraints) are relaxed [Geoffrion, 1974]. Alternatively, a polyhedral approach (also called *Branch & Cut*) improves the linear relaxation by iterative addition of new constraints that eliminate non-integral solutions [Hoffman and Padberg, 1985]. A specification of the polyhedral approach for *mixed binary programs* is given in [Balas *et al.*, 1993].

6.1.2 Branch & Bound

Branch & Bound uses a *tree representation* to implicitly explore the entire solution space $S = \{x \mid Ax = b, Dx \leq d, x \in \mathbb{N}^n\}$. Each node n of the tree corresponds to a subspace $S_n \subset S$. In the traditional Branch & Bound subspaces are defined by additional linear constraints:

$$S_n : \quad x \in S \quad (6.20)$$

$$A_n x = b_n \quad (6.21)$$

$$D_n x \leq d_n \quad (6.22)$$

$$(6.23)$$

The root of the tree represents the entire space of feasible solutions $S \subset \mathbb{R}^n$. The subspace S_c corresponding to a child node c is contained in the subspace S_p of the parent node p . Moreover, for each two children c_1 and c_2 of the same parent node the subspaces S_{c_1} and S_{c_2} have an empty intersection. In brief, for each node parent p we have:

$$S_p = S_{c_1} \sqcup S_{c_2} \quad (6.24)$$

where c_1 and c_2 are children of the node p .

At each node n of the tree the algorithm considers the optimization problem on a restricted solution space:

$$(\text{ILP}_n) : \quad \min c^T x \quad (6.25)$$

$$x \in S_n \quad (6.26)$$

A lower bound l_n for the optimum of the integer linear program ILP_n can be obtained from the linear relaxation. On the other side, when the algorithm treats a node the *best known feasible* solution $x^* \in S$ provides an upper bound u on the value of the optimal solutions. Crucially, if $u \leq l_n$ further exploration of the node can't lead to an improvement in the value of the objective function. In such case the node is *pruned*, i.e. eliminated from the succeeding consideration.

Conversely, if $l_n < u$ the subspace S_n may contain a solution that is better than x^* . If, in addition, the solution x_n^{LP} of the linear relaxation is *integral* (or, equivalently, $x_n^{LP} \in S_n$) it is precisely the solution of the integer linear problem (ILP_n). In such case we update the upper bound to $u = l_n$ and the best known solution to $x^* = x_n^{LP}$.

The linear relaxation in a node n is an optimization problem (LP- n) over a continuous set $L_n \subset \mathbb{R}^n$. If the node can't be pruned and the solution x_n^{LP} of (LP- n) contains non-integral elements we have to further explore the subspace S_n .

Here comes the *branching*.

Essentially, in the branching step we expand the tree by adding two children c_1 and c_2 to the node n . We use additional linear constraints to partition the subspace $S_n = S_{c_1} \sqcup S_{c_2}$ in subsets

$$S_{c_i} : \quad x \in S_n \quad (6.27)$$

$$P_{c_i}(x) = 1 \quad (6.28)$$

where $P_{c_i} : \mathbb{R}^n \rightarrow \mathbb{B}$ is some predicate. Traditionally, the predicate P_{c_i} is an inequality constraint.

The intention behind the branching is to *eliminate* the non-integral solution x_n^{LP} . More precisely, we chose the predicates P_{c_1} and P_{c_2} in a way that linear relaxations L_{c_1} and L_{c_2} of the sets S_{c_1} and S_{c_2} *don't contain* the partial value x_n^{LP} , i.e.

$$L_{c_1} \sqcup L_{c_2} \subsetneq L_n \text{ while } S_{c_1} \sqcup S_{c_2} \equiv S_n \quad (6.29)$$

Traditionally, for problems over binary variables the branching divides the solution space on subspaces corresponding to $x_i = 0$ and $x_i = 1$ for some variable x_i that has a fractional value in the relaxed solution x_n^{LP} . If a general integer variable $x_i \in \mathbb{N}$ gets the fractional value \tilde{x}_i , the conventional branching approach creates the nodes corresponding to $x_i \leq \lfloor \tilde{x}_i \rfloor$ and $x_i \geq \lceil \tilde{x}_i \rceil$.

The Branch & Bound tree exploration is illustrated on the figure 6.1.

We highlight that *the good quality bounds are essential for an efficient pruning*. The pruning is extremely important as the runtime of the algorithm heavily depends on the ability to eliminate suboptimal nodes.

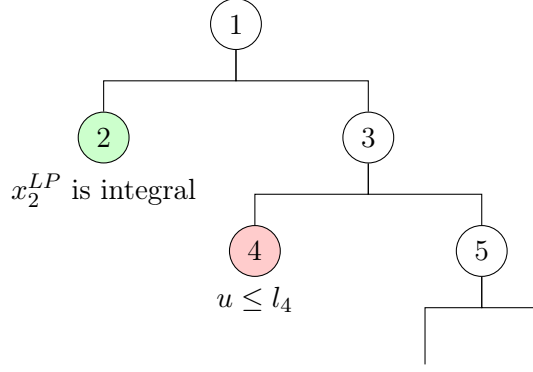


Figure 6.1: Branch and bound tree. At each node n one of three situations happens. If $u \leq l_n$ the node is pruned. Otherwise, if the solution of the relaxed problem x_n^{LP} is integral and better than the actual best solution the upper bound is updated: $x^* = x_n^{LP}$, $u = l_n$. Finally, if $l_n < u$ and x_n^{LP} has non-integral entries the node is branched.

6.2 Classical Branch & Price

We aim to address integer linear programs with a hybrid quantum-classical algorithm. For this purpose, we suggest a hybrid algorithm with a classical part that essentially reproduces the *Branch & Price* technique developed in [Vanderbeck, 1994]. Branch & Price is well-suited to address ILP with many variables. In Branch & Price a huge number of variables is tackled with the *column generation* approach that allows to consider only a restricted set of variables at each moment.

6.2.1 Dantzig-Wolfe decomposition

Column generation was originally introduced in [Ford and Fulkerson, 1958] to solve large linear programs over continuous variables arising in relaxations of the *path-flow* formulation of the *network-flow* problem. Column generation is particularly useful for (LP) instances when only few constraints in the standard formulation act on many variables. These constraints are referred as *coupling*. In what follows we assume that in the formulation (LP) the coupling constraints are given by the equalities $Ax = b$.

The (LP) instances are well-suited for the column generation if most constraints ($Dx \leq d$) act only on small subsets of variables. If, in addition, the matrix D has a block structure:

$$D = \begin{pmatrix} D_1 & & \\ & \ddots & \\ & & D_k \end{pmatrix} \quad (6.30)$$

the problem is particularly adapted for the *decomposition*. We denote by m_i and l_i the number of columns and rows respectively in each D_i .

To reason about the decomposable problem we introduce the notations:

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix}, \quad d = \begin{pmatrix} d_1 \\ \vdots \\ d_k \end{pmatrix}, \quad A = (A_1 \quad \dots \quad A_k) \quad (6.31)$$

where x_i and c_i are columns with m_i elements, $d_i \in \mathbb{R}^{l_i}$ and $A_i \in \mathbb{R}^{m_i \times l_i}$.

The decomposition principle [Dantzig and Wolfe, 1960] relies on the fact that every point in the *bounded convex space* $S = \{x \in \mathbb{R}^m \mid Dx \leq d\}$ can be represented as a convex combination of its *extreme points* $\{p^q\}_{q \in Q}$:

$$x = \sum_{q \in Q} \lambda^q p^q, \quad \sum_{q \in Q} \lambda_q = 1 \quad (6.32)$$

Representing the vectors $x_i \in \{y \in \mathbb{R}^{m_i} \mid D_i y \leq d_i\}$ by convex combinations we can obtain an equivalent formulation for (LP) that is called the *master program*:

$$(\text{M-LP}) : \quad \min \sum_{i=1}^k c_i^T \left(\sum_{q \in Q_i} \lambda_i^q p_i^q \right) \quad (6.33)$$

$$\sum_{i=1}^k A_i \left(\sum_{q \in Q_i} \lambda_i^q p_i^q \right) = b \quad (6.34)$$

$$\sum_{q \in Q_i} \lambda_i^q = 1, \quad \forall i \in [1, \dots, k] \quad (6.35)$$

$$\lambda_i^q \geq 0, \quad \forall i \in [1, \dots, k], \quad q \in Q_i \quad (6.36)$$

where $\{p_i^q\}_{q \in Q_i}$ are extreme points of the sets $S_i = \{y \mid D_i y \leq d_i\}$.

If we denote $c_i^q = c_i^T p_i^q$ and $a_i^q = A_i p_i^q$ the master program can be written as :

$$\min \sum_{\substack{i \in [1, \dots, k] \\ q \in Q_i}} c_i^q \lambda_i^q \quad (6.37)$$

$$\sum_{\substack{i \in [1, \dots, k] \\ q \in Q_i}} a_i^q \lambda_i^q = b \quad (6.38)$$

$$\sum_{q \in Q_i} \lambda_i^q = 1, \quad \forall i \in [1, \dots, k] \quad (6.39)$$

$$\lambda_i^q \geq 0, \quad \forall i \in [1, \dots, k], \quad q \in Q_i \quad (6.40)$$

The number of variables in (M-LP) is $N = |Q_1| + \dots + |Q_k|$. In practice N is very large. We highlight that in the approach that we present in the following the huge number of variables is not a problem as we never *explicitly* consider all variables together. Instead, the set of variables is given *implicitly* by the constraints $Dx \leq d$. To resume, using the decomposition we obtained for a linear program (LP) an equivalent formulation (M-LP) that has less constraints but much more variables.

6.2.2 Decomposition for integer linear programs

Now let's consider the ILP problem:

$$(\text{ILP}) \quad \min \quad c^T x \quad (6.41)$$

$$Ax = b \quad (6.42)$$

$$Dx \leq d \quad (6.43)$$

$$x \in \mathbb{N}^n \quad (6.44)$$

where (6.42) are the coupling constraints and the matrix D has a block-diagonal structure (6.30).

A bounded set $S_i = \{y \in \mathbb{N}^{m_i} \mid Dy \leq d\}$ of integer points can be explicitly enumerated:

$S_i = \{p_q\}_{q \in Q_i}$. Each part x_i of an integer vector $x = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$ can be trivially written as

$$x_i = \sum_{q \in Q_i} \lambda_i^q p_i^q \text{ where } \sum_{q \in Q_i} \lambda_i^q = 1, \lambda_i \in \{0, 1\}^{|Q_i|} \quad (6.45)$$

Similarly to the continuous problem, we use decomposition to rewrite the (ILP) formulation in an equivalent *master integer program* (M-ILP):

$$\text{(M-ILP) : } \min \sum_{\substack{i \in [1, \dots, k] \\ q \in Q_i}} c_i^q \lambda_i^q \quad (6.46)$$

$$\sum_{\substack{i \in [1, \dots, k] \\ q \in Q_i}} a_i^q \lambda_i^q = b \quad (6.47)$$

$$\sum_{q \in Q_i} \lambda_i^q = 1, \quad \forall i \in [1, \dots, k] \quad (6.48)$$

$$\lambda_i^q \in \{0, 1\}, \quad \forall i \in [1, \dots, k], q \in Q_i \quad (6.49)$$

where the difference with (M-LP) is the integrality condition (6.49). We also remark that for continuous case the set $\{p_q\}_{q \in Q}$ is made out of *extreme points* of the polyhedron $\{y \in \mathbb{R}^m \mid Dy \leq d\}$ while for ILP the set $\{p_q\}_{q \in Q}$ represents *all points* in $\{y \in \mathbb{N}^m \mid Dy \leq d\}$.

We highlight that the formulations (ILP) and (M-ILP) have the same integral solution value. However, their linear relaxations in general are *not equivalent*. Indeed, the very benefit of considering the extensive formulation (M-ILP) is due to the fact that sometimes the linear relaxation of (M-ILP) is *tighter*. In other words, if ν^* denotes the optimal value of (ILP) and, equivalently, (M-ILP) while the respective linear relaxations have the values ν_{ILP} and ν_{M-ILP} the following inequalities hold:

$$\nu_{ILP} \geq \nu_{M-ILP} \geq \nu^* \quad (6.50)$$

When an integer program is solved with Branch & Bound the tightness of the bounds is extremely important for an efficient elimination of suboptimal subspaces. Therefore, decomposition may lead to a more tractable formulation.

6.2.3 ILP formulations with large number of variables

Most optimization problems admit multiple ILP formulations. For instance, the *graph coloring* problem on $G = (V, E)$ can be naturally formulated using $K + |V|K$ binary variables:

$$\text{(GC-c) } \min \sum_{k=1}^K y_k \quad (6.51)$$

$$\sum_{k=1}^K x_i^k = 1, \quad \forall i \in V \quad (6.52)$$

$$x_i^k + x_j^k \leq 1, \quad \forall (i, j) \in E \quad (6.53)$$

$$x_i^k \leq y_k, \quad \forall i \in V, k \in [1, \dots, K] \quad (6.54)$$

$$x_i^k \in \{0, 1\}, y_k \in \{0, 1\}, \quad \forall i \in V, k \in [1, \dots, K] \quad (6.55)$$

where K is some upper bound on the chromatic number $\chi(G)$. The variable x_i^k is equal to 1 if the node i is assigned a color k . The variable y_k indicates if the color k is used. The constraint (6.52) requires each node to be colored and (6.53) assures that the coloring is *proper*.

Such formulation has multiple disadvantages that make it difficult for the Branch & Bound. Firstly, the linear relaxation provides a very poor bound $l = 1$ (achieved by the partial solution $x_i^k = 1/K, y_i = 1/K$). Secondly, the branching is not able to eliminate *symmetric* solutions that can be obtained with a simple permutation of colors. Therefore, the branching rarely improves the relaxed solution.

Alternatively, we can represent a proper coloring as a partition of nodes in independent sets: $V = S_1 \sqcup \dots \sqcup S_k$. If we take one binary variable y_s per independent set $s \in I(G)$ the optimization problem can be written as follows:

$$(GC-s) \quad \min \sum_{s \in I(G)} y_s \quad (6.56)$$

$$\sum_{\substack{s \in I(G) \\ v \in s}} y_s \geq 1, \quad \forall v \in V \quad (6.57)$$

$$y_s \in \{0, 1\}, \quad \forall s \in I(G) \quad (6.58)$$

The constraint (6.57) ensures that each node is contained in at least one independent set selected for the resulting partition. We remark that in general case the number of independent sets $|I(G)|$ in a graph is exponential on $|V|$.

A variety of *set covering* problems can also be formulated as ILP with many variables [Vanderbeck, 1994]. In set covering problems we have to select an ensemble of subsets of a set of *ground elements*. Depending on the particular problem, special constraints allow to decide if a subset is *appropriate* for selection. The objective is to find a ensemble of valid subsets that cover each element of the ground set.

Interestingly, certain QUBO instances can be also encoded in ILP formulations suitable for decomposition. For such instances a procedure solving QUBO with column generation was presented in [Mauri and Lorena, 2012].

6.2.4 Column generation

In previous sections we described different situations that lead to (ILP) with large amount of variables. Motivated by the terminology used in Dantzig-Wolfe decomposition we refer to such formulations as *master programs* (MP). In general, a master program is written in canonical form as follows:

$$(MP) : \quad \min \sum_{y \in S} c_y \lambda_y \quad (6.59)$$

$$\sum_{y \in S} a_y \lambda_y \geq b \quad (6.60)$$

$$\lambda_y \in \{0, 1\}, \quad \forall \lambda_y \in S \quad (6.61)$$

where the set of variables S is large. In this section we consider an approach that is specifically designed for such type of formulations. We remark that this approach remains tractable as it never considers a whole set of variables as once.

In practical applications the variables S are implicitly given by a *compact formulation*.

Example: In the set covering formulation for the graph coloring (GC-s) problem the set of variables S is the ensemble of stables $I(G)$.

If the master program is a result of decomposition, there is in S one variable per every point of the bounded set $\{y \in \mathbb{N}^n \mid Dy \leq d\}$. The compact formulation also indicates the values for the coefficients $c_y \in \mathbb{R}$ and $a_y \in \mathbb{R}^n$.

A *linear relaxation of the master problem* (LMP) is

$$(LMP) : \quad \min \sum_{y \in S} c_y \lambda_y \quad (6.62)$$

$$\sum_{y \in S} a_y \lambda_y \geq b \quad (6.63)$$

$$\lambda_y \leq 1, \quad \forall y \in S \quad (6.64)$$

$$\lambda_y \geq 0, \quad \forall y \in S \quad (6.65)$$

Dual problem

Each bound $\lambda_y \leq 1$ (6.64) corresponds to one dual variable π_y in the dual problem for (LMP). We use the notation $\boldsymbol{\pi}$ for the vector of m dual variables corresponding to coupling equality constraints (6.63). The *dual problem* (D-LMP) for (LMP) is formulated as:

$$(D-LMP) : \quad \max b^T \boldsymbol{\pi} - \sum_{y \in S} \pi_y \quad (6.66)$$

$$a_y^T \boldsymbol{\pi} - \pi_y \leq c_y, \quad \forall y \in S \quad (6.67)$$

$$\pi_y \geq 0, \quad \forall y \in S \quad (6.68)$$

$$\boldsymbol{\pi} \geq \mathbf{0}, \quad \boldsymbol{\pi} \in \mathbb{R}^m \quad (6.69)$$

We refer to the appendix A.1 for more details about the duality for linear programs.

If both primal and dual problems are bounded, the *strong duality theorem* states that their optimum values are equal [Gass and Harris, 2001]. Moreover, a *feasible* solution of the primal problem $\boldsymbol{\lambda}^*$ correspond to a *feasible* solution $\boldsymbol{\pi}^*$ of the dual problem *if and only if* the primal solution is *optimal*.

When the set of variables S is too large, the master problem can't be directly solved by conventional methods for ILP. Indeed, due to an exponential amount of variables even the linear relaxation is intractable. To tackle this issue, the work [Vanderbeck, 1994] adopts the *column generation* approach originally introduced for the cutting stock problem [Gilmore and Gomory, 1963].

Simplex method for large formulations

Column generation allows to solve the *linear relaxation* of the *master problem* (LMP) by considering a restricted set of variables S' . New variables are added to the set only if they can

improve the solution. Promising variables are selected using the *dual values* of the restricted linear relaxation.

The procedure is inspired by the *simplex method* [Nash, 2000] that we briefly present in the appendix A.2. We remark that primal and dual problems are simultaneously solved by the simplex method.

The simplex method operates a set B of *basis variables* of cardinality m where m is the number of inequality constraints in the canonical formulation. Remaining variables F are called *free variables*. At each iteration the method consider a *basic feasible solution* that is obtained by setting all free variables to zero. If a non-zero value of a free variable $\lambda_i \in F$ improves the value of the objective function, the basis is updated with a *pivot* operation. The pivot operation modifies the basis $B \rightarrow B'$ by replacing a carefully selected *leaving variable* $\lambda_e \in B$ with the promising free variable.

Successive basis modifications drives the method to a basic feasible solution that is optimal for (LP). Crucially, in an optimum obtained that way only m basis variables have non-zero values. Motivated by this observation, *column generation* adds the variables in a formulation only if they may appear in the optimum basis.

Pricing subproblem

At each iteration t column generation solves a restricted relaxation of master problem (t-LMP):

$$(t\text{-LMP}) : \quad \min \sum_{y \in S^t} c_y \lambda_y \quad (6.70)$$

$$\sum_{y \in S^t} a_y \lambda_y \geq b \quad (6.71)$$

$$\lambda_y \leq 1, \quad \forall y \in S^t \quad (6.72)$$

$$\lambda_y \geq 0, \quad \forall y \in S^t \quad (6.73)$$

where $S^t \subset S$. We denote by y^* the optimal solution for (t-LMP) and by (π_y^*, π^*) the corresponding values of the dual variables.

An *optimal* solution for the restricted problem (t-LMP) can be extended to a *feasible* solution for the initial (LMP) by setting to zero all variables λ_y for $y \in S \setminus S^t$. Similarly, we may obtain an assignment (not necessarily feasible) for the values of dual variables for (LMP). In the assignment for dual variables we take $\pi_y = 0$, $\forall y \in S \setminus S^t$. We can check if the *extended primal solution* is *optimal* for (LMP) by verifying if the *extended dual solution* is *feasible* for (DMP) (i.e. it satisfies the constraints 6.67).

For bounded problems the dual solution is infeasible if and only if there exists at least one primal variable $y' \in S \setminus S^t$ such that the inequality (6.67) is violated, i.e. $a_{y'}^T \pi^* > c_{y'}$. Such variable can be found by solving the problem:

$$\max a^T(y) \pi^* - c(y) \quad (6.74)$$

$$y \in S \setminus S^t \quad (6.75)$$

called *pricing subproblem (PS)*. For each variable $y \in S$ we refer to the value $r_y = a_y^T \pi^* - c_y$ as *reduced cost*.

When the optimum solution y' of the pricing subproblem has a value that is smaller or equal to zero the extended dual solution is feasible. Therefore, the optimum solution for the

restricted problem (t-LMP) is the same as the optimum of the original relaxation (LMP) over the entire set of variables. Otherwise, the variable found by (PS) can improve the value of the objective function. In such case column generation extends the set S^t with the new variable: $S^{t+1} = S^t \cup \{y'\}$ and the process restarts. The classical column generation is illustrated on the green part of the figure (6.2).

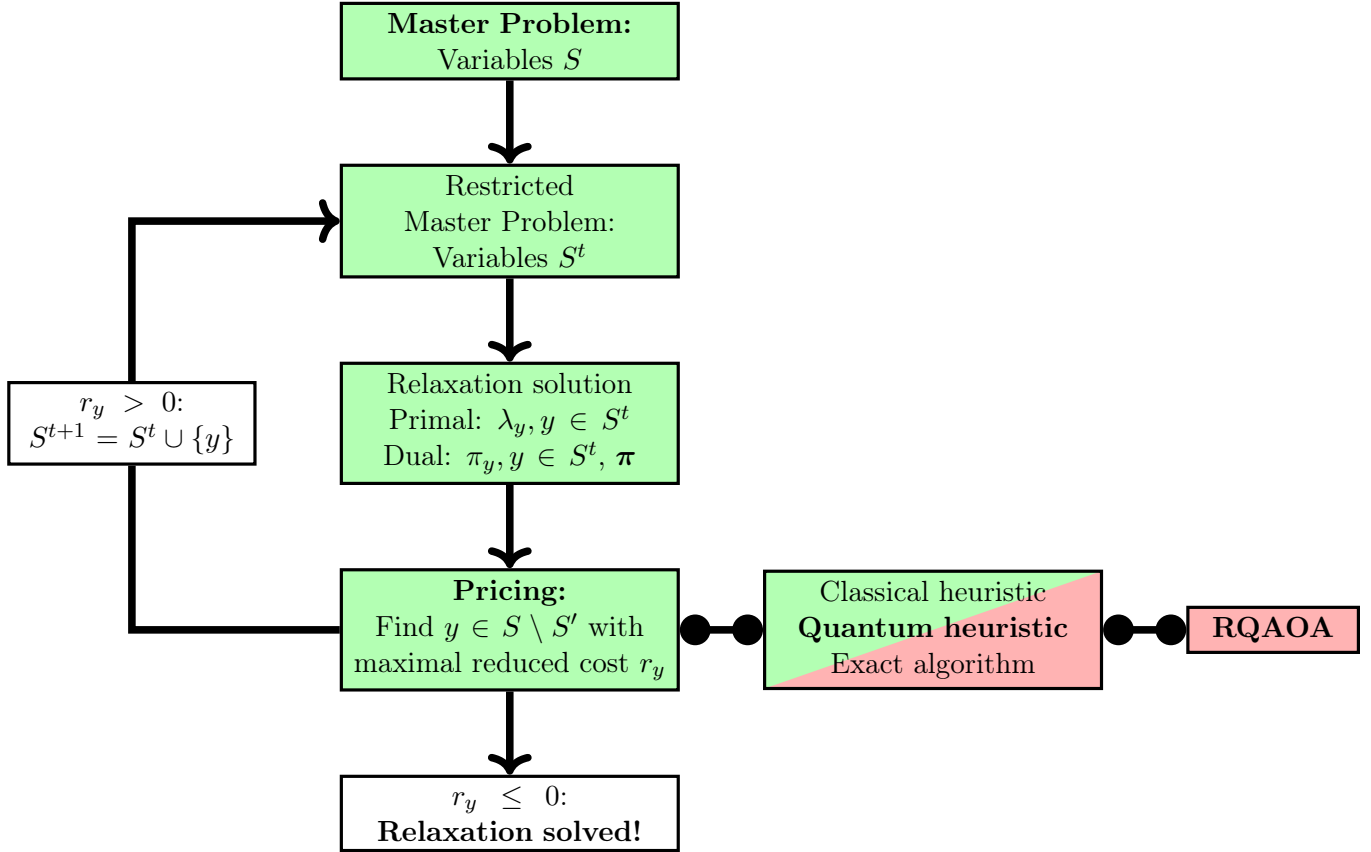


Figure 6.2: The quantum-classical column generation for huge linear relaxations. The relaxation is solved at each node of the Branch & Bound tree. The green nodes corresponds to the classical column generation and the red ones - to integrated quantum routines.

6.2.5 Branch & Price for Integer Linear Programs

Column generation is an approach that handles large linear optimization problems over *continuous variables*. The *Branch & Price* technique originally presented in [Vanderbeck, 1994] integrates column generation in Branch & Bound algorithm to derive an exact algorithm for large *integer linear programs*.

Branch & Price is a method for the formulations over many integer variables. Such formulations appear either naturally or as a result of decomposition of some compact (ILP) formulation. Just as Branch & Bound, Branch & Price implicitly explores the entire solution space in a tree search fashion. It uses the linear relaxation to bound the values of the objective function in each node of the tree. Nodes that can't lead to an improved solution are pruned from the tree.

The major distinction of the Branch & Price from the Branch & Bound consists in the way

they manage variables. During the entire execution Branch & Bound explicitly consider *all variables* of the problem. Branch & Price, on the contrary, operates a large set S of *implicitly* given variables. Generally, S is represented by a set on inequalities $Dp \leq d$ on elements p of a vector space with some finite dimension Q . Each integer point $p \in \mathbb{N}^Q$ satisfying the inequalities corresponds to one variable $y_p \in S$. Coefficients c_p and a_p for the variable y_p are usually given by some (often linear) functions $c : \mathbb{N}^Q \rightarrow \mathbb{R}$ and $a_p : \mathbb{N}^Q \rightarrow \mathbb{R}^m$ evaluated in the corresponding point p .

Example: An ensemble of independent sets $I(G)$ of a graph $G = (V, E)$ corresponds to points $x \in \{0, 1\}^{|V|}$ satisfying the inequalities $x_u + x_v \leq 1$ for every edge $(u, v) \in E$. In the set covering formulation for the graph coloring problem cost function coefficients for every variable are equal to one: $c \equiv 1$. The coefficients in covering constraints a_x are given by the characteristic vector x of the independent set.

In each node n of the branching tree the method solves the linear relaxation of some (ILP) problem over variables S . It begins by considering a restricted set of variables S^0 such that the relaxed problem admits a feasible solution on S^0 . The set of variables is iteratively extended $S^t \rightarrow S^{t+1} = S^t \cup \{y_p\}$ by the column generation that solves the *pricing subproblem*:

$$\text{Find } p \tag{6.76}$$

$$a_p^T \pi_t^* > c_p \tag{6.77}$$

$$Dp \leq d \tag{6.78}$$

$$p \in \mathbb{N}^Q \tag{6.79}$$

where $a_p = a(p)$, $c_p = c(p)$ and π_t^* is the dual solution of the restricted linear program.

The variable generation continues until the linear relaxation is solved to optimality. Equivalently, the generation stops when the *pricing subproblem* becomes infeasible. Thereafter if the relaxed solution is non-integral and the bound doesn't permit pruning the branching occurs (see figure 6.3).

Ryan-Foster branching rule

Due to the additional variables that join the formulation in the column generation process, traditional branching techniques are not adapted for the Branch & Price. Indeed, if we create two branches corresponding to $\lambda_y = 0$ and $\lambda_y = 1$ it is unclear how to ensure that the pricing problem doesn't generate the same variable again. Moreover, in many cases the constraint $\lambda_y = 0$ is very weak. For instance, in set covering problems it eliminates only one subset from an exponential number. Therefore, more sophisticated techniques are required that take in account the structure of the pricing problem.

When the master problem is a variant of the set covering problem we can use the *Ryan-Foster rule* [Ryan and Foster, 1981]. The rule creates two branches. In the first branch we require two elements x_1 and x_2 from the ground set to be covered by the same subset. In the second branch the opposite condition is imposed: valid subset can't simultaneously contain both variables.

Example: For the *graph coloring* problem the Ryan-Foster rule admits a natural interpretation. In this problem the *ground set* is the set of nodes V and variables correspond to independent sets. The rule firstly selects two nodes v and u that are not connected in the graph $(u, v) \notin E$. In the first branch the exploration continues on the variables λ_y that assign u and v to the same

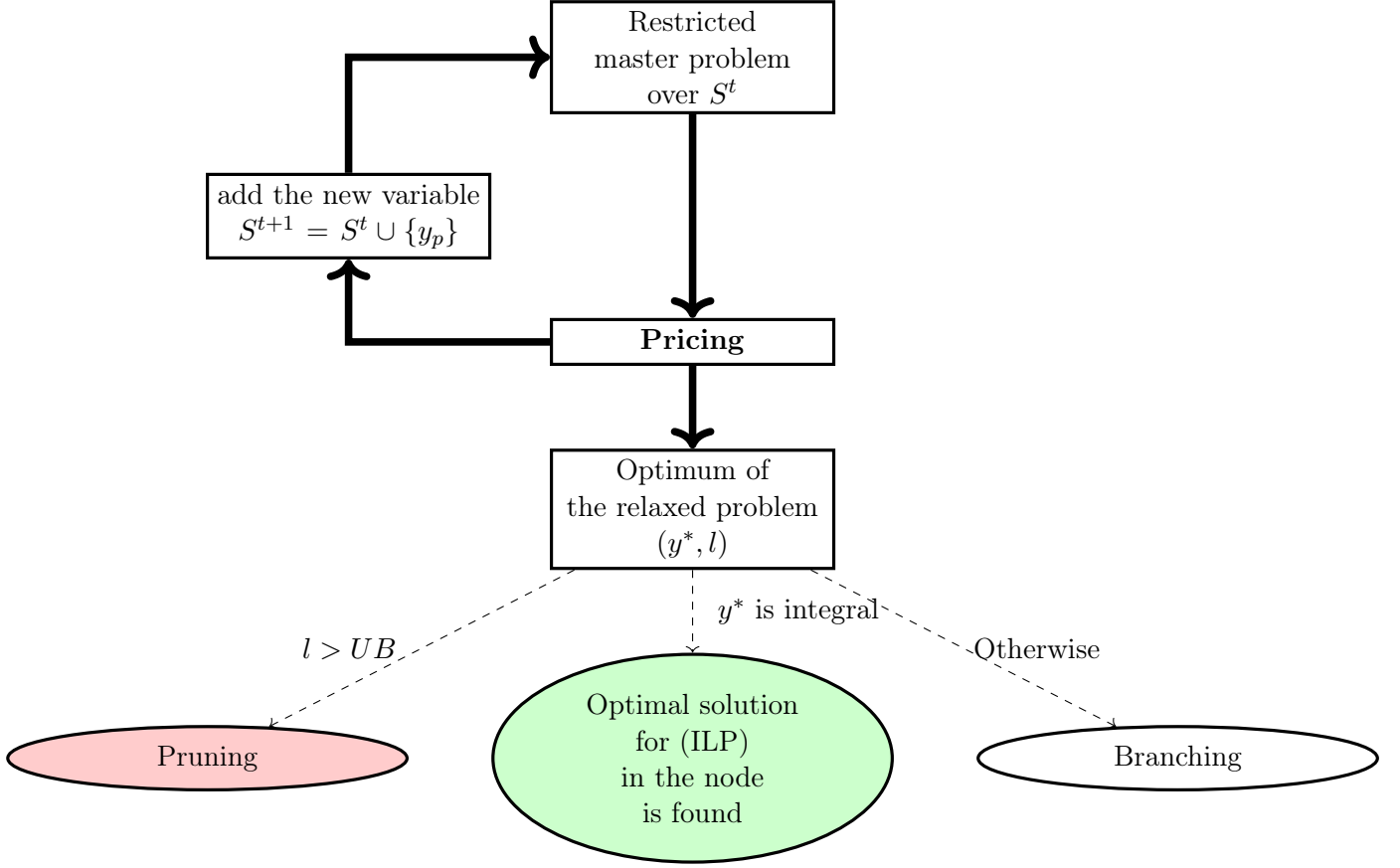


Figure 6.3: Branch & Price. The column generation stops when the pricing subproblem becomes infeasible. Then the method proceeds as traditional Branch & Bound.

color ($u \in y \iff v \in y$). This constraint may be communicated to the pricing subproblem by "merging" the nodes v and u in the graph. In the second branch the separation is enforced by addition of the new edge (u, v) to the graph.

6.3 Quantum-assisted Branch & Price

6.3.1 Embedding (R)QAOA into the Branch & Price

A hybrid method presented in [Svensson *et al.*, 2021] use QAOA as an heuristic solver *for the master problem* when it corresponds to a set covering problem.

Instead, we suggest a *different approach* that integrates quantum heuristics in the *pricing step* of the column generation (see fig. 6.2). From the dual solution π , the pricing problem (PS) is typically formulated as:

$$\max \pi^T A p - c_p \quad (6.80)$$

$$D p \leq d \quad (6.81)$$

$$p \in \mathbb{N}^Q \quad (6.82)$$

Such formulation is an *integer linear program* and, thus, NP-hard in general. In some cases (PS) has a particular structure that make it tractable for polynomial algorithms. For instance, in

the *path-flow* formulation of the *network flow* problem the optimum for the pricing subproblem corresponds to the shortest path in a graph [Desrosiers and Lübbecke, 2006]. More generally, if (PS) satisfy the *integrality property*⁷ it can be efficiently solved with the simplex method.

However, in most cases (PS) is an NP-hard problem.

Example: in the Branch & Price for *graph coloring* the pricing subproblem searches for an improving independent set. Each independent set s has a cost function coefficient $c_s = 1$ and it contributes to a covering constraint for every node $v \in s$. We denote by $x \in \{0, 1\}^{|V|}$ the incidence vector of a subset of V . Then (PS) can be formulates as follows:

$$\max \sum_{v \in V} \pi_v x_v \quad (6.83)$$

$$x_u + x_v \leq 1, \quad (u, v) \in E \quad (6.84)$$

$$x \in \{0, 1\}^{|V|} \quad (6.85)$$

which exactly corresponds to NP-hard *maximum weighted independent set* problem on the graph $G = (V, E)$ with node weights $w_v = \pi_v$.

When (PS) is NP-hard, finding an exact solution may take a long time. However, in Branch & Price we don't need precisely the exact solution but rather an answer to the question if there exists an improving variable. A variable y_p improves the solutions of the relaxation if it violates dual feasibility, i.e. if $\pi^T A p - c_p > 0$. Therefore, if an approximate solution \hat{p} for (PS) has the value $\nu_{\hat{p}} = \pi^T A \hat{p} - c_{\hat{p}} > 0$ we can accept it as a solution for (PS). For hard problems a sufficiently good *heuristic algorithm* can rapidly generate such variables.

On the other side, the algorithm ceases to generate columns only when there is no more improving variables. To verify if this termination condition is satisfied, one has to assure that the exact optimum value ν^* of (PS) is lower or equal to zero. Therefore, at each node of the branching tree we have to launch the exact algorithm for (PS) at least once to confirm that the relaxed problem is solved to optimality.

Taking in account previous remarks, we suggest a quantum-classical approach to efficiently solve the pricing subproblem. Our method proceeds in three steps. Firstly, it tries to find a solution with a classical heuristic. If the value of the obtained approximate solution is $\nu_c > 0$ we declare the (PS) solved and add the corresponding variable to the formulation. Otherwise, we launch a *quantum heuristic*. If both heuristics fail, an exact algorithm is called. The exact algorithm either finds an improving variable or guarantees the optimality of the relaxed solution.

For the graph coloring example the resulting hybrid algorithm for the pricing subproblem is sketched on Figure 6.4. A scheme for the overall Quantum-assisted Branch & Price technique is given on Figure 6.2.

6.4 Graph Coloring with Quantum-assisted Branch & Price

We illustrate our hybrid approach on the *graph coloring* problem .

The graph coloring problem searches for a proper vertex coloring with minimal number of colors. The optimal value of the problem is called *chromatic number* $\chi(G)$ of the graph. In an extensive formulation of the coloring problem we aim to minimize the number of independent sets required to cover every vertex.

⁷Extreme points of the polyhedron $Dp \leq d$ are integral

Finding a minimal graph coloring is NP-hard [Garey and Johnson, 1990]. The decision version of the problem called k -colorability belongs to Karp's 21 NP-complete problems [Karp, 1972]. For k -colorability the asymptotically best exact algorithm based on dynamic programming has the complexity $O(2.4423^n)$ [Lawler, 1976].

In addition, minimal graph coloring is NP-hard to approximate within a factor $n^{1-\epsilon}$ for all $\epsilon > 0$ [Zuckerman, 2007]. The best known approximation algorithm achieves the approximation ratio of $O(n(\log \log n)^2(\log n)^3)$ [Halldórsson, 1993].

Graph coloring is used to model multiple real-world problems from different domains such as scheduling, resource allocations and many others [Marx, 2003, Lewis, 2021]. Therefore, efficient algorithms for this problem are of extreme importance. Naive approaches are based on greedy ideas that color vertices following some order [Brélaz, 1979]. We refer to [Husfeldt, 2015] for a complete review of diverse classical algorithms for the graph coloring.

Quantum heuristics can be directly applied to the problem if some QUBO formulation is provided. One such formulation was presented in [Kochenberger *et al.*, 2005]. Interestingly, a generic *tabu search* algorithm for QUBO was able to find competitive solutions on this last formulation.

More refined quantum approach results from the *Quantum Alternating Operator Ansatz* framework. An implementation presented in [Hadfield *et al.*, 2019] encodes the color of each node in a sequence of $D_G + 2$ qubits where D_G is a maximum node degree in the graph. The quantum system is initialized in a trivial feasible solution that uses $D_G + 1$ colors. Mixer operator is tailored to preserve the evolution in the subspace of proper colorings.

6.4.1 Column generation

An algorithm for the *graph coloring* based on column generation was originally presented in [Mehrotra and Trick, 1996]. This algorithm considers a set-covering formulation (GC-s)

$$\text{(GC-s):} \quad \min \sum_{s \in I(G)} \lambda_s \quad (6.86)$$

$$\sum_{\substack{s \in I(G) \\ v \in s}} \lambda_s \geq 1, \quad \forall v \in V \quad (6.87)$$

$$\lambda_s \in \{0, 1\}, \quad \forall s \in I(G) \quad (6.88)$$

This formulation contains one variable per independent set. In general case, the number of variables in (GC-s) is exponential.

The optimum value of the problem is equal to the chromatic number of the graph $\chi(G)$. Adopting the notation from [Held *et al.*, 2012], we use $\chi_f(G)$ to refer to the optimum of the linear relaxation also called *fractional chromatic number*. Clearly, $\chi_f(G)$ provides a lower bound of the chromatic number. Although there exists efficient algorithms for linear programs over continuous variables, the value $\chi_f(G)$ is also NP-hard to approximate with a factor $n^{1-\epsilon}$ for all $n > \epsilon$ [Zuckerman, 2007]. The complexity of the relaxed problem is consistent with the exponential amount of variables in the formulation.

Column generation is used to optimize the linear relaxation of (GC-s), i.e. to evaluate $\chi_f(G)$. It may be further integrated in an exact coloring algorithm implementing the Branch & Price approach.

The state-of-the-art program *exactcolor* solving the graph coloring problem with Branch & Price was presented in [Held *et al.*, 2012]. In this program the initial set of variables S^0 for the

restricted master problem is obtained by the *DSatur* heuristic [Brélaz, 1979]. In the column generation a generic LP-solver [Gurobi Optimization, LLC, 2022] optimizes the linear relaxation over a restricted set of variables S^t . The LP-solver returns values for dual variables π_v , $v \in V$ that define the pricing subproblem (MWIS):

$$(\text{MWIS}) : \quad \max \sum_{v \in V} \pi_v x_v \quad (6.89)$$

$$x_u + x_v \leq 1, \quad (u, v) \in E \quad (6.90)$$

$$x_v \in \{0, 1\}, \quad v \in V \quad (6.91)$$

The generation halts when the optimum independent set s^* found by the pricing problem has weight $\pi(s^*) = \sum_{v \in s^*} \pi_v \leq 1$. If the pricing subproblem returns a solution with $\pi(s^*) > 1$ the restricted set of variables S^t is extended by s^* .

The *exactcolor* program solves the pricing subproblem with a two-step approach. Firstly, it search for an heuristic solution with a method that combines greedy initialization with 2-interchange local search. If the heuristic fails to find an improving variable, an exact Branch & Bound algorithm is used that either returns a promising variable or confirms that it doesn't exist.

If the linear relaxation has non-integral solution the node is branched according to the *Ryan-Foster* rule.

Numerical experiments confirming the interest of this approach were reported in [Held *et al.*, 2012]. In one of these experiments column generation was used to evaluate the fractional chromatic number $\chi_f(G)$. On 119 out of 136 instances from the DIMACS benchmark [Johnson and Trick, 1996] the program terminated in less then three days. The fractional chromatic number computed by the program improved previously known bounds for 5 instances.

6.4.2 A hybrid approach to the pricing subproblem

We suggests a hybrid method that adopts a combined quantum-classical approach for the pricing MWIS subproblem. Our motivation lies in good results that quantum heuristics were able to achieve on MWIS (see section 5.3.3).

The work [Svensson *et al.*, 2021] suggests a different combination of quantum and classical routines in the Branch & Price framework. In their approach column generation extends the set of variables until the linear relaxation is solved to optimality. The resulting variables form a set $S' \subset S$. In the next step a *quantum heuristic* is used to computes an *approximate integer solution* for the restricted master problem over S' . We remark that when the master problem can be interpreted as a *set packing* problem it admits a natural QUBO formulation [Alidaee *et al.*, 2008].

Contrary to our approach, this second technique doesn't naturally integrate in the traditional Branch & Price framework. Indeed, even an exact integer optimum on the restricted set S' isn't necessary equal to the optimum over S . In other words, although the variables from $S \setminus S'$ are null in some optimum of the relaxed problem, they can have positive values in the optimal integer solution. Therefore, the only information that we can obtain with a quantum heuristic applied to the restricted master problem is a new upped bound u on the optimal solution. This information is practical if, for instance, u matches the rounded relaxed optimum: $u = \lceil l_{S'} \rceil \equiv \lceil l_S \rceil$ or if it improves the best known upper bound.

In addition, as the set S' grows when new variables are generated the restricted master problem can rapidly become *intractable* for the quantum hardware of limited size. We remark that in our approach quantum heuristics are always called on a *fixed number* of variables $|V|$.

In our hybrid technique the instance of MWIS problem that appears at the pricing step is tackled with a three-fold procedure that combines the classical heuristic from [Held *et al.*, 2012], RQAOA and the exact algorithm (see fig 6.4).

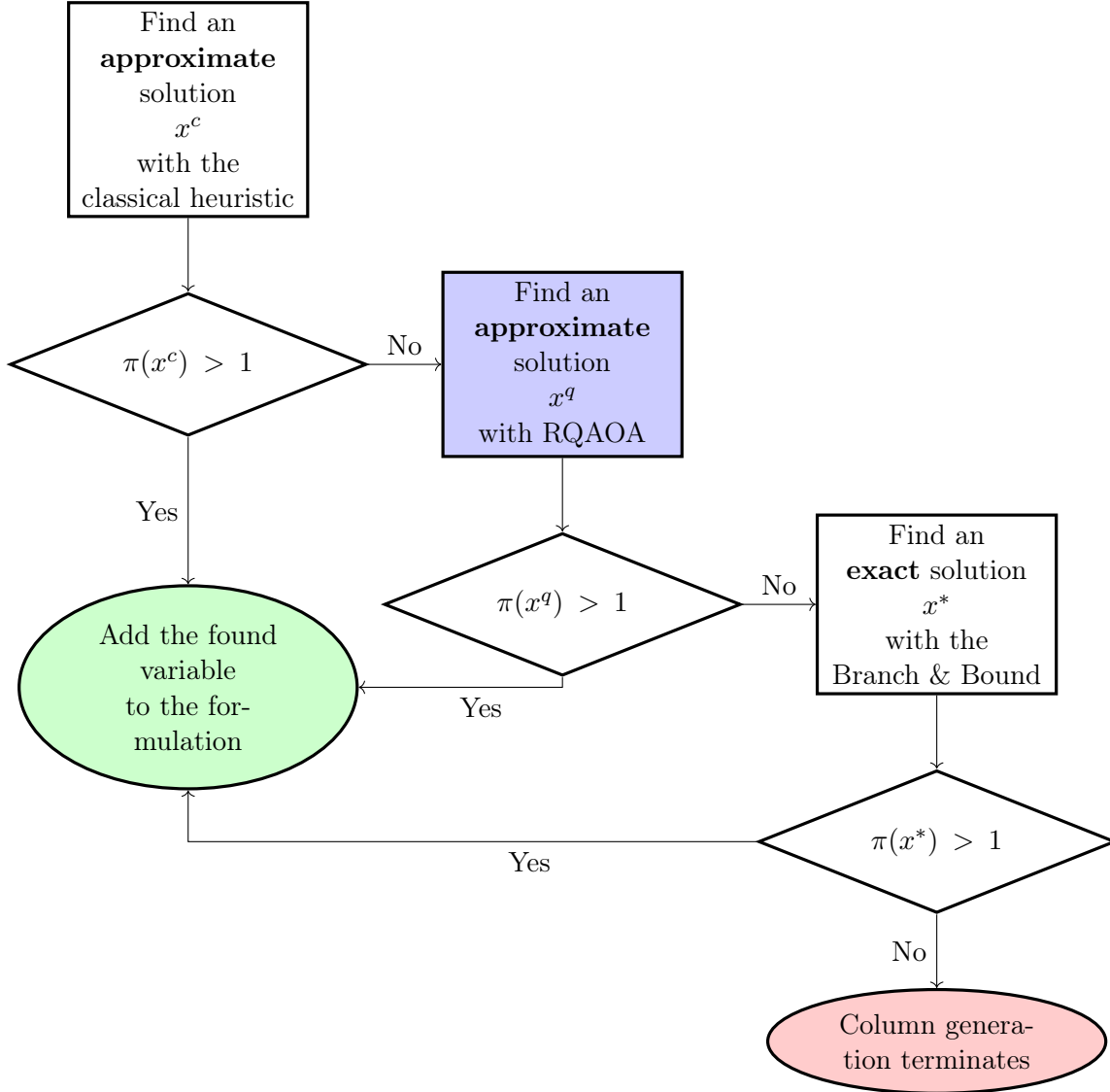


Figure 6.4: Hybrid algorithm for the pricing subproblem for graph coloring. In our implementation, we use the same classical heuristic and Branch & Bound implementation as in [Held *et al.*, 2012].

Different execution scenarios for the pricing step are described in the table 6.1.

We define the *RQAOA improvement rate* as a ratio of pricing calls in which the classical heuristic failed but the RQAOA succeeded (green line in the table 6.1) to the total amount of cases when there were an improving solution but the classical heuristic failed (green and red lines in the table 6.1).

Quantum algorithms are expected to improve over classical heuristics only in some special cases [McClean *et al.*, 2021]. If the pricing encounters one of such instances, the use of quantum heuristic allows to postpone the necessity to execute a costly exact algorithm. Therefore, in the

Classical heuristic	Quantum heuristic	Solution exists (call exact algorithm)
Succeeded	-	True
Failed	Succeeded	True
Failed	Failed	True
Failed	Failed	False

Table 6.1: Possible execution scenarios of the pricing step

quantum heuristic it sufficiently fast, the overall procedure is accelerated.

6.4.3 Numerical results

We evaluated the *RQAOA improvement rate* on 10 Erdos-Renyi random graph with 50 nodes.

Density	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Classical heuristic success rate	0.95	0.84	0.76	0.92	0.89	0.95	0.96	1	1
RQAOA improvement rate	0.75	0.10	0.03	0.07	0.04	0	0.2	—	—

Table 6.2: Classical heuristic success rate (the fraction of improving variables found by the classical heuristic algorithm) and RQAOA improvement rate.

This table shows that it is indeed profitable for some instances to use the hybrid method instead of the traditional classical approach. We recall that the RQAOA is launched only if the classical heuristic fails to find the improving variable, i.e. only on rather difficult instances.

To test our approach in realistic settings we considered instances issued from a modified version of the Charge Task Selection problem described in chapter ??(5). In the new coloring version the conflict graph remains the same, while the new objective requires to cover *all charges* with a minimal number of stations. Previously (see section 5.3.3) we have reported that RQAOA sometimes finds better approximation than the classical heuristic for *unweighted MIS* on the instances issued from the Charge Task Selection problem.

Unfortunately, for the instances of the *coloring version* of the Charge Task Selection problem we've observed that there is no need for RQAOA as the classical heuristic always finds the solution of the pricing problem whenever it exists at all. A spectacular performance of the classical heuristic compared to our previous result is probably due to the presence of *weights* that have a special structure determined by the dual solution of the relaxation.

Variable results on different instances indicates that the performance enhancement resulting from the integration of quantum heuristics is not robust across different families of instances.

6.5 Discussion

In this chapter we presented a hybrid quantum-assisted Branch & Price method for large integer linear programs. In principle, it can be adapted to *any* formulation over huge set of variables that is implicitly given by another pricing integer program.

However, the additional cost related to the integration of quantum heuristics should be justified by two conditions:

- the computational difficulty of the pricing subproblem for classical algorithms

- amenability of the pricing subproblem to a QUBO formulation

These conditions are fulfilled by the set covering formulation of the graph coloring problem. Another example is the *cutting stock* problem for which the pricing subproblem is equivalent to 0-1 knapsack problem [Gilmore and Gomory, 1963]. The 0-1 knapsack problem was addressed with QAOA in [de la Grand'rive and Hullo, 2019].

In addition, if the formulation is a result of decomposition for block-diagonal D the size of resulting k disconnected pricing subproblems is small compared to the size of the compact formulation. It may be that the compact formulation has too many variables but the subproblems are small enough to be handled on near-future quantum hardware.

Part III

Variational algorithms with ZX calculus

Chapter 7

Introduction to ZX calculus

ZX-calculus, originally introduced in [Coecke and Duncan, 2011], is a *graphical language* that allows to reason about quantum computing. In this language complex computations on qubits are represented with *diagrams*. Diagrams are made out of *elementary generators*. Each diagram corresponds to a linear transformation between qubit spaces. In a sense, a ZX-diagram is a *tensor network* representation of a linear map. A compact set of *rewrite rules* allows to transform diagrams into equivalent ones. The notable advantage of ZX-calculus compared to other representation (including linear maps, circuits and tensor networks) is that computations may be done *entirely graphically*. In other words, we can manipulate matrices of exponential size using local transformations on more compact diagrammatic representation.

The work [van de Wetering, 2020] provides an excellent introduction to ZX-calculus together with a fairly detailed review of important results. This introduction assumes a general background in quantum computing. A much more exhaustive book [Coecke and Kissinger, 2017] introduces the entire field of quantum computing directly in the language of ZX-diagrams. A *Python package PyZX* [Kissinger and van de Wetering, 2020a] provides automatic tools to reason about ZX-diagrams. It is particularly useful to study diagrams that are too large to be analyzed by hand. Many works on ZX-diagrams, including this thesis, use the *Tikzit* software⁸ to draw ZX-diagrams.

ZX-calculus was used to address many quantum applications. For instance, it was used to establish important results for the *measurement-based quantum computing (MBQC)* [Duncan and Perdrix, 2010, Kissinger and van de Wetering, 2019]. We recall that MBQC is an alternative model for quantum algorithms. In MBQC the computation is performed by properly designed measurements of a special quantum state. A detailed introduction to MBQC is available in [Jozsa, 2005].

Other than MBQC, ZX-calculus was successfully applied for circuit optimization [Cowtan *et al.*, 2020, de Beaudrap *et al.*, 2020, Kissinger and van de Wetering, 2020b, Duncan *et al.*, 2020]. The graphical language is also adapted for design and verification of quantum error correction codes [Garvie and Duncan, 2018]. Moreover, ZX-calculus is particularly well-suited for the analysis and compilation of *surface codes* [Horsman, 2011, de Beaudrap and Horsman, 2020, Hanks *et al.*, 2020]. Surface codes are extremely promising for the fault-tolerant quantum computing.

In the context of combinatorial optimization an extension of ZX-calculus called ZH-calculus was used to reason about the respective complexities of *constraint satisfaction problem (CSP)* and $\#CSP$ (the maximization of a number of satisfied clauses) [de Beaudrap *et al.*, 2021]. In the work [de Beaudrap *et al.*, 2021] both problems CSP and $\#CSP$ on the same input were represented by the same diagram, while with interpretation over different rings. A polynomial

⁸<https://tikzit.github.io>

complexity of 2-SAT is reflected in the simplicity of the graphical reduction of the corresponding ZH-diagram.

Another extension of ZX-calculus called ZXH-calculus was used to study physical properties of 1D AKLT state [East *et al.*, 2022]. Interestingly, AKLT states are *ground states* of quantum many-body systems under a multidimensional Heisenberg spin model.

Finally, ZX-calculus also found an application in variational quantum algorithms. In particular, the *barren plateau* phenomena for parameterized quantum circuits, i.e. the exponential vanishing of the gradient with the number of qubits, can be conveniently analyzed with the graphical tools [Zhao and Gao, 2021, Toumi *et al.*, 2021]. The work [Fontana *et al.*, 2020] use ZX-diagrams to express the symmetries in the parameter landscape.

7.1 ZX diagrams

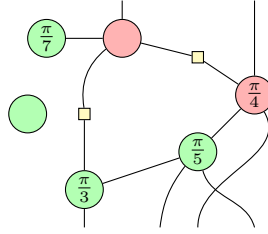


Figure 7.1: Example of a ZX-diagram

As we can see on the figure 7.1, a ZX-diagram is a network made out of nodes and wires. There are three types of nodes: red spiders $\textcircled{\alpha}$, green spiders $\textcircled{\alpha}$ and Hadamard boxes \square . Red and green spiders sometimes are drawn with a rational number inside. This number is called *phase* or *angle*. Both spiders can have an arbitrary number of adjacent wires. The Hadamard box has strictly two wires. A wire can connect two nodes or, alternatively, have free ends that point either up or down. Wires with free ends pointing towards the top or towards the bottom of the diagram are called *inputs* and *outputs* respectively. We denote by $D(n, m)$ or $D : n \rightarrow m$ a diagram with n inputs and m outputs. Diagrams with no inputs *or* no outputs are called *states* and *effects* respectively. If a diagram has no input *and* no output wires it is called a *scalar*.

In quantum computing ZX-diagrams are used as a graphical representation of linear maps. A diagram with n inputs and m outputs corresponds to a complex matrix with 2^n columns and 2^m rows. Using the terminology from category theory we call *interpretation functor* (or, alternatively, a standard interpretation) the map $\llbracket \cdot \rrbracket$ that associates to a ZX-diagram the corresponding matrix:

$$\llbracket D(n, m) \rrbracket = M_{2^m \times 2^n}(\mathbb{C}) \quad (7.1)$$

7.1.1 Generators

Formally, ZX-diagrams are inductively defined from a set of *basic generators* combined with sequential and parallel compositions.

The generator $\llbracket \cdot \rrbracket : 0 \rightarrow 0$ is an empty ZX-diagram. It correspond to the scalar value 1. For more complex ZX-diagrams the elementary building blocks are red and green spiders and the hadamard box.

Spiders

The *green spider* also called *Z-spider* is a diagram with $n \in \mathbb{N}$ inputs and $m \in \mathbb{N}$ outputs with the interpretation:

$$\left[\begin{array}{c} n \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ m \end{array} \right] = |0 \dots 0\rangle \langle 0 \dots 0| + e^{i\alpha} |1 \dots 1\rangle \langle 1 \dots 1| \quad (7.2)$$

The spider is parameterized by a real number $\alpha \in \mathbb{R}$ called *phase*. As the parameter α is used in a complex power of an exponent, we can restrict the range to $\alpha \in [0, 2\pi]$. We remark that the linear map that corresponds to the Z-spider is a *symmetric tensor*.

The *red spider* is alternatively referenced as X-spider. Like the green spider, it can have an arbitrary number of inputs $n \in \mathbb{N}$ and outputs $m \in \mathbb{N}$. The red spider is also decorated by a real value α that we can take in the range $\alpha \in [0, 2\pi]$.

$$\left[\begin{array}{c} n \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ m \end{array} \right] = |+\dots+\rangle \langle +\dots+| + e^{i\alpha} |-\dots-\rangle \langle -\dots-| \quad (7.3)$$

In this work we use the popular convention that the phase $\alpha = 0$ can be omitted.

Z-spider can be used to express some well-known states and matrices such as *Z-rotation gate* and vectors $|+\rangle$ and $|-\rangle$ of the *Hadamard basis*:

$$\left[\begin{array}{c} \alpha \\ | \\ | \end{array} \right] = |0\rangle \langle 0| + e^{i\alpha} |1\rangle \langle 1| = R_Z(\alpha) \quad (7.4)$$

$$\left[\begin{array}{c} | \\ | \end{array} \right] = |0\rangle + |1\rangle = \sqrt{2}|+\rangle \quad (7.5)$$

$$\left[\begin{array}{c} \pi \\ | \\ | \end{array} \right] = |0\rangle - |1\rangle = \sqrt{2}|-\rangle \quad (7.6)$$

With X-spider we can draw the diagrams for the vectors from the computational basis $\{|0\rangle, |1\rangle\}$. Red spiders with one input and one output corresponds to the X-rotation with angle α .

$$\left[\begin{array}{c} \alpha \\ + \\ + \end{array} \right] = |+\rangle \langle +| + e^{i\alpha} |-\rangle \langle -| = R_Z(\alpha) \quad (7.7)$$

$$\left[\begin{array}{c} + \\ + \end{array} \right] = |+\rangle + |-\rangle = \sqrt{2}|0\rangle \quad (7.8)$$

$$\left[\begin{array}{c} \pi \\ + \\ + \end{array} \right] = |+\rangle - |-\rangle = \sqrt{2}|1\rangle \quad (7.9)$$

It follows from the definition that the no-input no-output spiders π and π correspond to the zero scalar.

Hadamard box

The Hadamard gate $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is represented by a yellow box diagram:

$$\llbracket \text{yellow box} \rrbracket = |+\rangle\langle 0| + |-\rangle\langle 1| \quad (7.10)$$

Later in this section we will see that the linear map H can alternatively be represented by a combination of green and red spiders.

Wire generators

A number of other useful linear transformations are expressed with wires. A simple wire with nothing on it corresponds to the identity:

$$\llbracket | \rrbracket = |0\rangle\langle 0| + |1\rangle\langle 1| \quad (7.11)$$

We remark that empty one-input one-output spiders corresponds to the same transformation:

$$\text{green spider} = \text{red spider} = | \quad (7.12)$$

In the expression (7.12) we use the equality notation $D_1 = D_2$ for diagrams to tell that the diagrams have the same semantics:

$$D_1 = D_2 \iff \llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket \quad (7.13)$$

In the following we will see that there are many cases where two visually different diagrams have the same matrix interpretation. The ZX-calculus provides a set of rules that allows to transform diagrams into semantically equivalent ones entirely in the graphical framework.

Another wire generator are the *cup* $\cup : 2 \rightarrow 1$, *cap* $\cap : 0 \rightarrow 2$ and SWAP $\times : 2 \rightarrow 2$. These generators are interpreted as:

$$\llbracket \text{cup} \rrbracket = |00\rangle\langle 00| + |10\rangle\langle 01| + |01\rangle\langle 10| + |11\rangle\langle 11| = \text{SWAP} \quad (7.14)$$

$$\llbracket \cup \rrbracket = \langle 00| + \langle 11| \quad (7.15)$$

$$\llbracket \cap \rrbracket = |00\rangle + |11\rangle \quad (7.16)$$

7.1.2 Compositions

Elementary generators are used to derive bigger diagrams using *sequential* and *parallel* composition. The parallel composition is usually referred as *tensor product*.

Tensor product

We can obtain a ZX-diagram by putting two other diagrams aside each other. Formally, for two diagrams $\begin{array}{|c|} \hline \dots \\ \hline D_0 \\ \hline \dots \end{array} : n \rightarrow m$ and $\begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \end{array} : k \rightarrow l$ we define their *tensor product* as a diagram

$$\begin{array}{|c|} \hline \dots \\ \hline D_0 \\ \hline \dots \end{array} \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \end{array} : n + k \rightarrow m + l.$$

The term "*tensor product*" is employed as the interpretation of the new diagram is chosen to be exactly the Kronecker product of the matrices corresponding to D_1 and D_2 :

$$\left[\begin{array}{|c|c|} \cdots & \cdots \\ \hline D_0 & D_1 \\ \hline \cdots & \cdots \end{array} \right] = \left[\begin{array}{|c|} \cdots \\ \hline D_0 \\ \hline \cdots \end{array} \right] \otimes \left[\begin{array}{|c|} \cdots \\ \hline D_1 \\ \hline \cdots \end{array} \right] \quad (7.17)$$

Sequential composition

As we have observed on our first example (7.1), the output wires of one generator are usually connected to input wires of other generators. This procedure is called *sequential composition*.

More precisely, for two diagrams $\left[\begin{array}{|c|} \cdots \\ \hline D_0 \\ \hline \cdots \end{array} \right] : n \rightarrow m$ and $\left[\begin{array}{|c|} \cdots \\ \hline D_1 \\ \hline \cdots \end{array} \right] : m \rightarrow l$ the diagram $\left[\begin{array}{|c|} \cdots \\ \hline D_0 \\ \hline \cdots \\ \hline D_1 \\ \hline \cdots \end{array} \right]$ is interpreted as

$$\left[\begin{array}{|c|} \cdots \\ \hline D_0 \\ \hline \cdots \\ \hline D_1 \\ \hline \cdots \end{array} \right] = \left[\begin{array}{|c|} \cdots \\ \hline D_1 \\ \hline \cdots \end{array} \right] \circ \left[\begin{array}{|c|} \cdots \\ \hline D_0 \\ \hline \cdots \end{array} \right] \quad (7.18)$$

where the composition $A \circ B$ of two matrices $A_{2^m \times 2^n}$ and $B_{2^l \times 2^m}$ is matrix multiplication:

$$A \circ B = AB \quad (7.19)$$

In the terminology of *tensor networks* each node of the diagram corresponds to a tensor and the wire connecting two nodes represents tensor contraction.

Only topology matters

Using the sequential composition and the tensor product we can compute the interpretation of complex diagrams such as the one on the figure 7.1. However, our current definition strictly discriminates between input and output wires. Therefore, the nodes of the diagram have to come with a predefined order. It implies, in particular, that we still can't interpret a *horizontal wire*.

It turns out that in ZX-diagram the respective order of nodes actually *doesn't matter*. In other words, we can freely move nodes on the 2D plane. As soon as the order of inputs and outputs is preserved, the interpretation doesn't change. This fantastic property follows from the set of equalities:

$$\begin{array}{c} \text{cup} = | = \text{cap} \quad \text{cross} = || \quad \text{self-loop} = \cap \quad \text{figure-eight} = \cup \end{array} \quad (7.20)$$

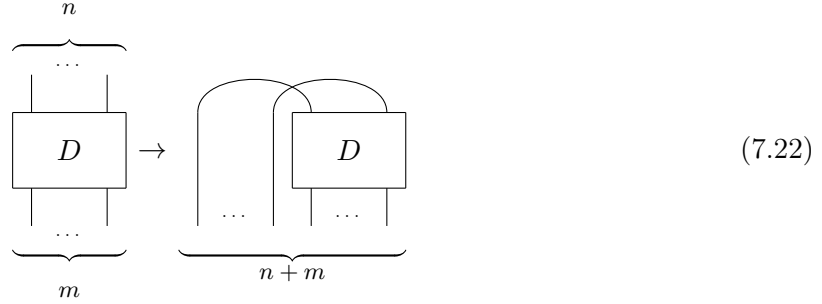
$$\begin{array}{c} \text{green circle with 4 wires} = \text{green circle with 4 wires} \quad \text{green circle with 2 wires} = \text{green circle with 2 wires} \quad \text{green circle with 1 wire} = \text{green circle with 1 wire} \end{array} \quad (7.21)$$

that are grouped under the paradigm "*Only topology matters*". We remark that the second set of formulas (7.21) is also true for the red spider.

These equalities can be verified by a careful computation of the corresponding matrices. Resulting matrix computations are remarkably tedious comparing to a simplicity of corresponding

transformations in ZX-calculus. In fact, in the graphical language we only have to remember that the wires can be bent at will.

By bending wires we can transform any diagram $D : n \rightarrow m$ with n inputs and m outputs to a state $D' : 0 \rightarrow n + m$:

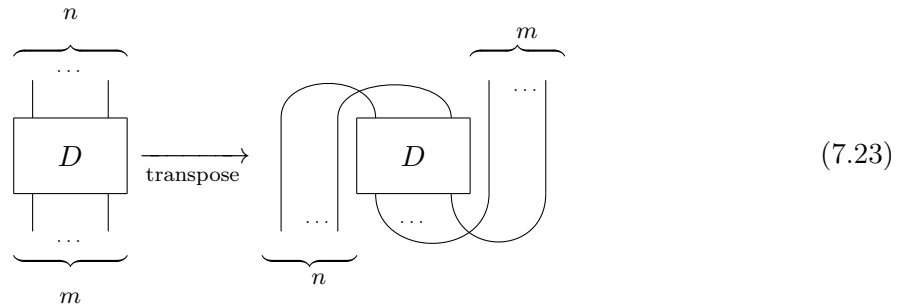


For fixed dimensions n and m such mapping is bijective. The mapping is commonly referred as the *Choi-Jamiołkowski* isomorphism [Haapasalo, 2019].

7.2 Adjoint diagram

If we replace all phases α in a diagram D by their negations the resulting diagrams D' will correspond to a matrix that is *conjugate* of $\llbracket D \rrbracket$. Indeed, for the complex phases $e^{-i\alpha} = \overline{e^{i\alpha}}$

Using cups and caps we can also find the *transpose* of a diagram:



By combining this two transformations we can obtain for each diagram its *adjoint* in a simple way. Moreover, as we can bend internal wires at will it actually is sufficient to simply "flip" the diagram upside down and negate the phases.

7.3 Universality

ZX-diagrams is a universal language for linear maps on dimensions that are powers of 2 [Coecke and Duncan, 2011]. The universality states that for each every linear map $M : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^m}$ there exists a diagram $D : n \rightarrow m$ such that the interpretation of D is exactly equal to M .

The universality is relatively easy to prove. Firstly, we observe that every complex number $c \in \mathbb{C}$ can be represented as a scalar diagram $D : 0 \rightarrow 0$. Indeed, from the definition of generators we have:

Lemma 7.3.1

$$\llbracket \text{green circle} \rrbracket = 2 \qquad \llbracket \text{green circle with } \alpha \rrbracket = 1 + e^{i\alpha} \qquad (7.24)$$

$$\left[\begin{array}{c} \text{red circle} \\ \text{green circle} \end{array} \right] = \sqrt{2} \quad \left[\begin{array}{c} \text{red circle with } \pi \\ \text{green circle with } \alpha \end{array} \right] = \sqrt{2}e^{i\alpha} \quad (7.25)$$

$$\left[\begin{array}{c} \text{red circle} \\ \text{green circle} \end{array} \right] = \frac{1}{\sqrt{2}} \quad (7.26)$$

Any complex number can be decomposed as $Z = e^{i\beta} \cos(\alpha)2^n$. The previous equation shows how this product can be obtained by composing scalar diagrams.

The universality of ZX-diagrams for unitary matrices straightforwardly follows from the universality of the gate set $\{H, \text{CNOT}, R_X(\theta), R_Z(\theta)\}$ for the pure qubit quantum computing. For H and $R_Z(\theta)$ the ZX-diagrams are already known. By explicitly computing the interpretation, we can show that:

$$\left[\begin{array}{c} \text{green circle} \\ \text{red circle} \end{array} \right] = \sqrt{2} \text{CNOT} \quad (7.27)$$

A universal gate set allows to compute any complex vector of unit norm starting from the trivial state $|0 \dots 0\rangle$. For other vectors we proceed by multiplication by the scalar factor equal to the norm. For maps with different number of input and outputs we use the Choi-Jamiołkowski isomorphism (7.22).

An alternative proof of universality of ZX-diagrams was obtained using the normal forms [Jeandel *et al.*, 2019].

We remark that while it is quite easy to pass from a circuit to a diagram the backward translation is much more challenging [Duncan *et al.*, 2020]. In fact, if a diagram doesn't correspond to a unitary matrix such translation is impossible without using ancilla and post-selection. In the opposite case, a circuit can be efficiently extracted for a diagram that has a *gflow* [Backens *et al.*, 2021].

Fragments

Up to now we considered the diagrams without any specific restriction on the phases. In other words, in general ZX-calculus the phases are allowed to take any value in $\mathbb{R}/2\pi\mathbb{Z}$. It turns out that in some settings it is meaningful to restrict the values of the angles to a specific sub-group \mathcal{G} of $\mathbb{R}/2\pi\mathbb{Z}$. In such case we speak about the fragments of the language, conventionally denoted as $\text{ZX}_{\mathcal{G}}$ -calculus.

The interpretation of a ZX-diagram with general angles is a matrix with elements in \mathbb{C} . It was proven in [Jeandel *et al.*, 2019] that for a specific fragment defined by a group \mathcal{G} the corresponding matrices have elements in the ring $\mathcal{R}_{\mathcal{G}} = \mathbb{Z} \left[\frac{1}{\sqrt{2}}, e^{i\mathcal{G}} \right]$. The notation $\mathbb{Z} \left[\frac{1}{\sqrt{2}}, e^{i\mathcal{G}} \right]$ stands for the smallest ring that contains \mathbb{Z} , $\frac{1}{\sqrt{2}}$ and $\{e^{i\alpha} \mid \alpha \in \mathcal{G}\}$. It was shown in [Jeandel *et al.*, 2019] that if the group \mathcal{G} contains $\frac{\pi}{4}$ the $\text{ZX}_{\mathcal{G}}$ -diagrams are universal for the matrices over $\mathbb{Z} \left[\frac{1}{\sqrt{2}}, e^{i\mathcal{G}} \right]$.

A particular interest is attracted to finitely generated fragments. We use the notation $\mathcal{G} = \langle x_1, \dots, x_n \rangle$ to denote the smallest sub-group of $\mathbb{R}/2\pi\mathbb{Z}$ that contains all x_i from the list. If the set of group generators contain only one element α , we will refer to the corresponding fragment as α -fragment

One of the most famous fragment is the Clifford fragment with $\mathcal{G} = \langle \pi \rangle$. Diagrams from the Clifford fragment allow to reason about the *stabilizer quantum mechanics* [Backens, 2014].

The stabilizer quantum mechanics studies all transformations that can be obtained using measurements in the computational basis and unitary circuits made out of elements of the Clifford group. A peculiar property of such transformations is that they can be efficiently simulated on a classical computer [Aaronson and Gottesman, 2004]. A slightly more powerful *real Clifford* $\frac{\pi}{2}$ -fragment was investigated in [Duncan and Perdrix, 2014].

Gates from the Clifford group completed with so-called T-gate $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ are approximately universal for pure qubit quantum computations. This set of gates can be represented by diagrams from the $\frac{\pi}{4}$ -fragment [Jeandel *et al.*, 2018a, Ng and Wang, 2018].

Other fragments over rational angles were discussed in [Jeandel *et al.*, 2019].

7.4 Rewrite rules

As we have seen before, ZX-diagrams can represent any linear map of dimension $2^m \times 2^n$. However, a diagrammatic representation is far from being unique. We've already mentioned in (7.12) an example of multiple diagrams for the trivial *identity* matrix.

A semantical equivalence of two diagrams can be established by explicitly computing corresponding matrices. Such approach is not very inspiring as the matrix representation usually has an exponential size for a given diagram. A slightly more efficient technique proceeds by the *tensor network contraction*. However, both approaches significantly reduce the interest of compact diagrammatic representation.

Alternatively, ZX-diagrams can be manipulated with semantic-preserving rewrite rules. A rule is an equality between diagrams $D_1 : n \rightarrow m$ and $D_2 : n \rightarrow m$ with the same amount of inputs and outputs. A rule (r) is called *sound* or, equivalently, *semantic-preserving* if:

$$D_1 \underset{(r)}{=} D_2 \iff \llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket \quad (7.28)$$

In principle, any sound equality $D_1 = D_2$ can be states as a rule. For practical reasons, we are interested in keeping the set of rewrite rules as small as possible. If such set of rules is sufficient to establish any semantically valid equality it is called *complete*. In the following we present the most important rewrite rules together with the motivation behind them.

7.4.1 Meta-rule

According to the inductive definition, a composition of two ZX-diagrams is another diagram. When a sound rewrite rule is used to transform one of parts of the diagram the interpretation of the overall diagram doesn't change. This means that the local application of sound rewrite rules preserves the semantics. We call this fact a meta-rule:

$$\begin{array}{c} \begin{array}{|c|} \hline \dots \\ \hline D_0 \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array} \implies \begin{array}{|c|} \hline \dots \\ \hline \begin{array}{|c|} \hline D_0 \\ \hline \dots \\ \hline \end{array} \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline \begin{array}{|c|} \hline D_1 \\ \hline \dots \\ \hline \end{array} \\ \hline \dots \\ \hline \end{array} \end{array} \quad (7.29)$$

In other words, if it is true that $\begin{array}{|c|} \hline \dots \\ \hline D_0 \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array}$ for two diagrams $D_1, D_2 : n \rightarrow m$ than for all

$D_2 : k \rightarrow l$ and $D_3 : m \rightarrow l$ we have:

$$\begin{array}{|c|} \hline \dots \\ \hline D_0 \\ \hline \dots \\ \hline \end{array} \begin{array}{|c|} \hline \dots \\ \hline D_2 \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array} \begin{array}{|c|} \hline \dots \\ \hline D_2 \\ \hline \dots \\ \hline \end{array} \text{ and } \begin{array}{|c|} \hline \dots \\ \hline D_0 \\ \hline \dots \\ \hline D_3 \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline D_3 \\ \hline \dots \\ \hline \end{array} \quad (7.30)$$

7.4.2 Hadamard rules

The Hadamard gate has two crucial properties. The first property, called Euler decomposition, states that the Hadamard gate can be obtained with a diagram made out of green and red spiders:

$$\begin{array}{|c|} \hline \dots \\ \hline \square \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline \text{red spider} \\ \hline \dots \\ \hline \end{array} \begin{array}{|c|} \hline \dots \\ \hline \text{green spider} \\ \hline \dots \\ \hline \end{array} \quad (EU) \quad (7.31)$$

In fact, there are multiple ways to represent a Hadamard with a fully spider diagram [van de Wetering, 2020]. We selected this particular one as it doesn't contain global scalars. In this representation it is also obvious that the Hadamard gate is equal to its own transpose. Moreover, together with some rewrite rules that we will see later (EU) allows to diagrammatically prove that the Hadamard gate is equal to its own inverse, i.e.:

Lemma 7.4.1 [[van de Wetering, 2020]]

$$\begin{array}{|c|} \hline \dots \\ \hline \square \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline \square \\ \hline \dots \\ \hline \end{array}$$

The second significant property of the Hadamard gate maps is that it maps the computational basis to the Hadamard basis:

$$H|0\rangle = |+\rangle \quad (7.32)$$

$$H|1\rangle = |-\rangle \quad (7.33)$$

This property translates to the following diagrammatic equality:

$$\begin{array}{|c|} \hline \dots \\ \hline \text{red spider} \\ \hline \dots \\ \hline \end{array} = \begin{array}{|c|} \hline \dots \\ \hline \text{green spider} \\ \hline \dots \\ \hline \end{array} \quad (H) \quad (7.34)$$

This equality is often referred as *Hadamard conjugation*. With the Hadamard conjugation rule we can flip colors of all red spiders and transform the diagram into a *graph state* form [Duncan and Perdrix, 2009]. Moreover, combined with the lemma (7.4.1) the *Hadamard conjugation* implies that *any equality between two diagrams remains true with flipped colors of spiders*.

It was shown in [Duncan and Perdrix, 2009] that the rule (EU) can't be derived from a simpler set of rules. On the other hand, the second rule (H) is particularly important to manipulate nodes of different colors. For this reasons we select (EU) and (H) as basic rewrite rules.

7.4.3 Spider fusion

The special structure of green and red spider implies multiple useful diagrammatic equalities. One of them, the *identity removal*, was already presented in (7.12). The identity removal states that we can remove an empty spider from a wire.

We have also seen that one can freely bend the legs of a spider (7.21). In fact, the symmetric structure of spiders leads to a much more powerful family of equalities called *spider fusion*:


$$(7.35)$$

where " $\cdot \cdot$ " means that there is at least one wire between spiders.

According to this rule we can *fuse* any two spiders of the same color that are connected by at least one wire. The phase of the resulting spider is equal to the sum of phases of individual spiders.

7.4.4 Bialgebra rules

In the previous section we have seen that two connected spiders of the *same color* can be fused. The specific interaction of two spiders of *different colors* follows from the fact that the *computational basis* $\{|0\rangle, |1\rangle\}$ and the *Hadamard basis* $\{|+\rangle, |-\rangle\}$ correspond to *complementary* observables Z and X . In quantum mechanics complementarity means that we can't simultaneously measure the value of two observables in a quantum state [Kiukas *et al.*, 2019].

We remark that the interpretation of Z-spider implies that the spider :

$$(7.36)$$

copies the states from the computational basis, i.e. for a vector $|b\rangle$, $b \in \{0, 1\}$ we have $\text{COPY } |b\rangle = |bb\rangle$. As the vector $|0\rangle$ is (up to a scalar) a one-legged red spider (7.8) we can write this equality as:

$$(7.37)$$

Using this rule together with (K), (S1) we can derive the following lemmas:


Lemma 7.4.2

$$(7.38)$$

Lemma 7.4.3

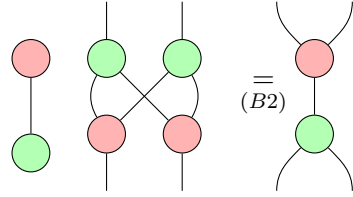
$$(7.39)$$

where k is a natural number.

In addition, we observe that the spider  applied to a basis state $|b_1 b_2\rangle$, $b_1, b_2 \in \{0, 1\}$ outputs their sum in F_2 : $|b_1 \oplus b_2\rangle$:



$$\left[\begin{array}{c} \text{spider} \end{array} \right] = |0\rangle\langle 00| + |0\rangle\langle 11| + |1\rangle\langle 01| + |1\rangle\langle 10| = \text{XOR} \quad (7.40)$$

An important rule about interaction of two spiders is a consequence of the relation of XOR and COPY operators:



$$(7.41)$$

Roughly speaking, this rule states that when we apply XOR to two entries and then copy the output we obtain the same result as when we start by copying each input and then XORing each pair separately.

In mathematical terms the rule (B2) states that the spider  and  form a *bialgebra*. A fundamental consequence of (B1) and (B2) (and fusion rules) is the *Hopf rule*:

Lemma 7.4.4

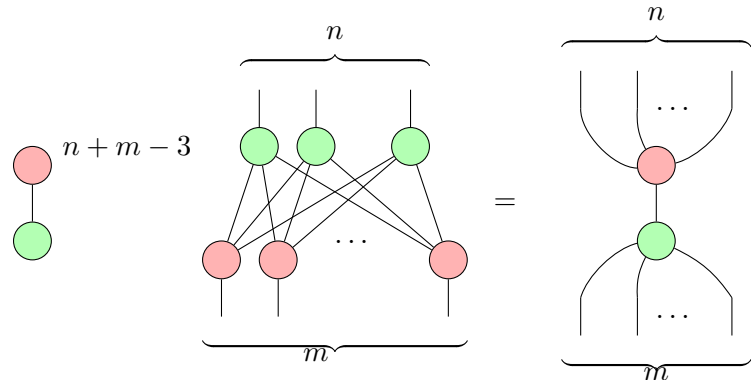


$$(7.42)$$

The Hopf rule is precisely the expression of *complementarity* of Z and X in the language of ZX-diagrams.

Finally, by iteratively applying the bialgebra rule we can obtain an equation for arbitrary number of interacting spiders:

Lemma 7.4.5 [Coecke and Kissinger, 2017], theorem 9.71



$$(7.43)$$

7.4.5 Completeness

Previously, we have introduced some important graphical rewrite rules that are *sound*:

$$D_1 \underset{(r)}{=} D_2 \implies \llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket \quad (7.44)$$

Using these rules we can perform computations diagrammatically, i.e. without passing by the matrix representation of the tensor network contraction. A set of rules \mathcal{A} is called *complete* if it allows to demonstrate that any two diagrams with the same semantics are equivalent, i.e.

$$\llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket \implies \exists (r_1), \dots, (r_n) \in \mathcal{A} \text{ s.t. } D_1 \underset{(r_1)}{=} \dots \underset{(r_n)}{=} D_2 \quad (7.45)$$

We use the term *axioms* to speak about the rules from a complete set \mathcal{A} .

For a long time the question about the existence of a compact set of complete rules was open. The first demonstration of completeness appeared for the Clifford fragment [Backens, 2014]. It was followed by a result for the real Clifford fragment [Duncan and Perdrix, 2014].

First complete set of rules for the approximately universal Clifford+T fragment was presented in [Jeandel *et al.*, 2018a]. It was followed by two completeness results for the general ZX-calculus [Hadzihasanovic *et al.*, 2018, Jeandel *et al.*, 2018b].

On the figure 7.2 we show a compact set of axioms presented in [Jeandel *et al.*, 2019]. This set of rules is complete for the $\frac{\pi}{4}$ -fragment. In [Vilmart, 2019] it was shown that only one additional axioms suffices to make such set of rules complete for general ZX-diagrams. This axiom is:

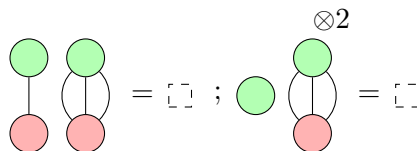


$$(7.46)$$

where the angles $\alpha_{i=\{1,2,3\}}$, $\beta_{j=\{1,2,3\}}$ and γ are related by a non-linear side condition. The additional axiom essentially states the equivalence of Euler decompositions of one-qubit unitary transformation.

Completeness is one of the major advantages of the ZX-calculus. It implies that we can prove any sound diagrammatic equality by a series of local graphical transformations from a compact set of axioms. We remark that in the set of rules on the Figure 7.2 we explicitly keep the scalar factors that are usually ignored in papers about ZX-calculus. The reason for it will become clear later when we will present the addition of diagrams. To easily manipulate scalars we will extensively use the lemma representing the equality $\frac{1}{\sqrt{2}} \times \sqrt{2} = 1$:

Lemma 7.4.6 ([Jeandel *et al.*, 2019])



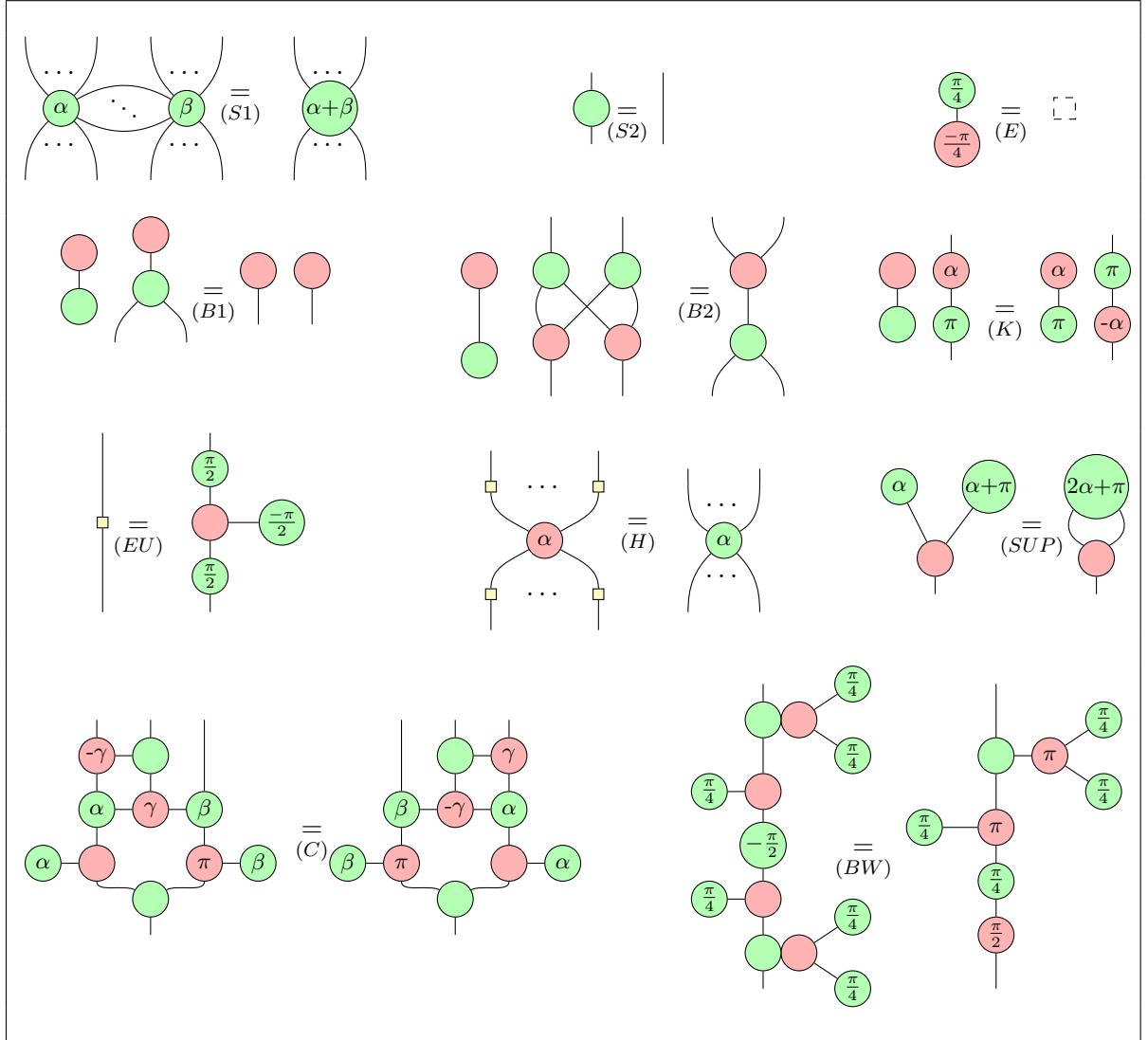


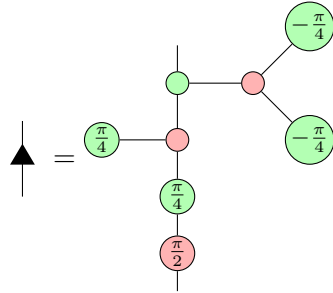
Figure 7.2: Axioms for ZX. All rules remain true flipped upside down and with inverted colors. Families of equations are given using ‘dots’: \dots means any number of wires, $\dot{\cdot}$ means at least one wire.

Triangle

Initially, completeness proofs used the translation from ZX-calculus to another language called ZW-calculus for which a complete set of rules already existed. More modern results [Vilmart, 2019, Jeandel *et al.*, 2019] follow a different approach that pass by *normal forms*. More precisely, in such approach any diagram is transformed to a specific normal form. The reduction is carried out in a finite number of steps. The normal form is exclusively determined by the underlying matrix. Thus, two diagrams with the same semantics are reduced to the same normal form.

The construction for the normal form extensively uses a *syntactic sugar* called *triangle* orig-

inally introduced in [Jeandel *et al.*, 2018a]:


(7.47)

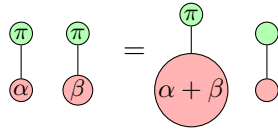
The triangle corresponds to a non-unitary transformation:

$$\left[\begin{array}{c} \uparrow \\ \downarrow \end{array} \right] = |0\rangle\langle 0| + |1\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad (7.48)$$

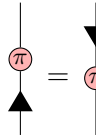
that maps $|0\rangle$ to $\sqrt{2}|+\rangle$ and $|1\rangle$ to $|1\rangle$.

The work [Jeandel *et al.*, 2019] proves multiple useful lemmas involving the triangle:

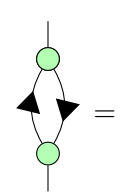
Lemma 7.4.7



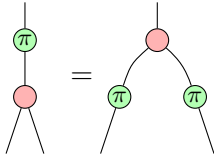
Lemma 7.4.13



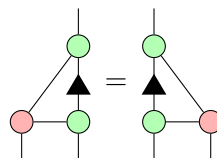
Lemma 7.4.18



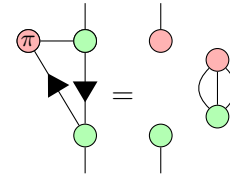
Lemma 7.4.8



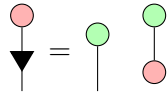
Lemma 7.4.14



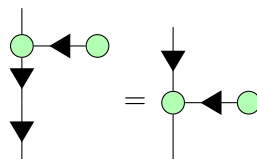
Lemma 7.4.19



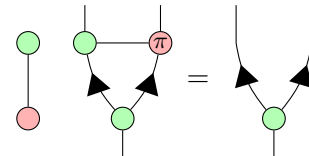
Lemma 7.4.9



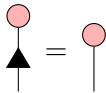
Lemma 7.4.15



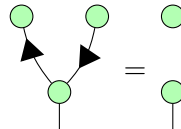
Lemma 7.4.20



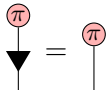
Lemma 7.4.10



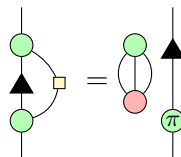
Lemma 7.4.16



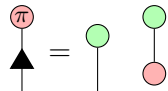
Lemma 7.4.11



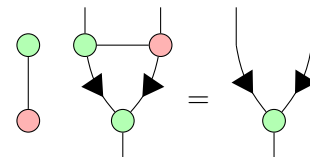
Lemma 7.4.17



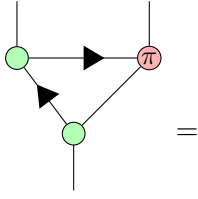
Lemma 7.4.12



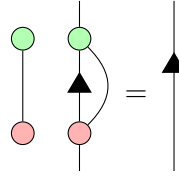
and



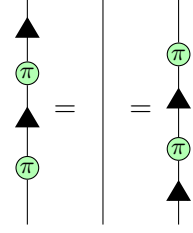
Lemma 7.4.21



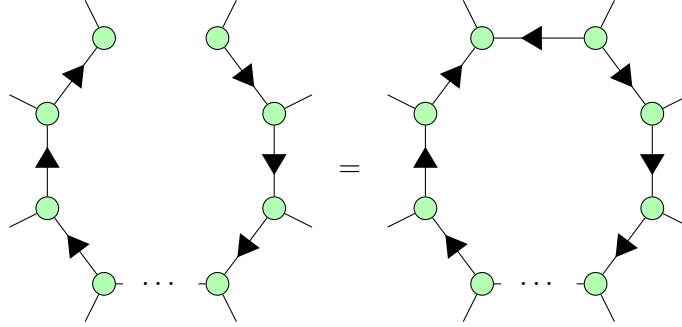
Lemma 7.4.22



Lemma 7.4.23

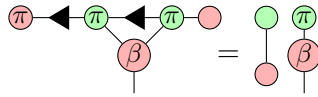


Lemma 7.4.24

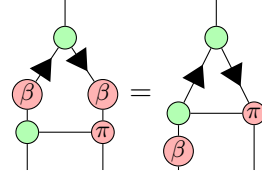


We also introduce some new lemmas that will be useful in the following chapters. The proofs serve as illustration of the graphical computation process.

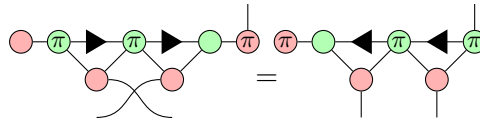
Lemma 7.4.25



Lemma 7.4.26



Lemma 7.4.27



Proof: [Proof (lemma 7.4.25)]

$$\begin{array}{c}
 \text{Diagram 1} \\
 \text{Diagram 2} \\
 \text{Diagram 3}
 \end{array}
 \stackrel{\substack{(B1) \\ 7.4.12 \\ 7.4.6}}{=}
 \begin{array}{c}
 \text{Diagram 4} \\
 \text{Diagram 5}
 \end{array}
 \stackrel{7.4.9}{=}
 \begin{array}{c}
 \text{Diagram 6} \\
 \text{Diagram 7}
 \end{array}
 \quad (7.49)$$

■

Proof: [Proof (lemma 7.4.27)] The left hand side is:

$$\begin{array}{c}
 \text{Diagram 1} \\
 \text{Diagram 2}
 \end{array}
 \stackrel{\substack{(B1) \\ 7.4.9}}{=}
 \begin{array}{c}
 \text{Diagram 3} \\
 \text{Diagram 4}
 \end{array}
 \quad (7.50)$$

For the right hand side we get:

$$(7.51)$$

Proof: [Proof (lemma 7.4.26)]

To prove the lemma we adopt a slightly different technique than a usual sequence of rewrites. The technique consists in verification that the diagrams from the left hand side and the right hand side lead to the same result when applied on a set of *basis states* $\{|0\rangle, |1\rangle\}$ corresponding to diagrams and . As was pointed out in [Coecke and Kissinger, 2017] for two matrices $M_1, M_2 : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^m}$ and a basis $\{|b_1\rangle, \dots, |b_{2^n}\rangle\}$ in \mathbb{C}^{2^n} the equality on basis input is equivalent to the equality of matrices, i.e.

$$\forall i \in [1, \dots, 2^n] : M_1|b_i\rangle = M_2|b_i\rangle \iff M_1 \equiv M_2 \quad (7.52)$$

Therefore, by demonstrating that = and = we actually establish that $\left[\begin{array}{c} \boxed{lhs} \\ \dots \end{array} \right] = \left[\begin{array}{c} \boxed{rhs} \\ \dots \end{array} \right]$. Finally, the lemma condition follows from the completeness of ZX-calculus.

$$(7.53)$$









$$(7.54)$$

7.5 Linear diagrams

In ZX-diagrams the angles inside spiders are real numbers or elements of some group \mathcal{G} . However, in many equations such as (S1), (K), (H), (SUP) and (C) we used *symbols* α and β instead of numbers. The symbolic notation was extremely handy to define *families of equalities*, i.e.

equalities that hold for any real values of angles. By explicitly keeping symbols inside spiders we can draw *parameterized diagrams*. A *parameterized diagram* with n different symbols can be interpreted as a function that associates to each element of the space \mathbb{R}^n a well-defined ZX-diagram.

Between parametrized diagrams, we distinguish a family of *linear diagrams* denoted $\text{ZX}(\beta)$:

[Linear diagrams [Jeandel *et al.*, 2018b]] A ZX-diagram is linear in β_1, \dots, β_k with constants in $L \subset \mathbb{R}$ if it is generated by , , , , , , ,  combined by tensor product and composition with α of the form $\sum_i n_i \beta_i + c$ with $n_i \in \mathbb{Z}$ and $c \in L$.

It was shown in [Jeandel *et al.*, 2018b] that for $L = \{\frac{n\pi}{4}\}_{n \in \mathbb{Z}}$ the Clifford+T axiomatization (7.2) is complete for linear diagrams.

We say that a diagram D is parametrized by β_1, \dots, β_k if its angles are some functions on β_1, \dots, β_k . We denote such a diagram by $D(\beta_1, \dots, \beta_k)$.

It is possible to *evaluate* a parametrized diagram $D(\beta)$, $\beta \in \mathbb{R}^k$ in a point $\beta^0 \in \mathbb{R}^k$ by replacing every occurrence of β_i with the respective value β_i^0 . The result of evaluation is a diagram $D(\beta_0)$ from $\text{ZX}_{\mathbb{R}}$.

The family of linear diagrams may appear restricted compared to ZX-diagrams that allow angles from a more general class of functions. It is, however, sufficient for applications in variational quantum algorithms as they use circuits where parameters appear in a linear fashion [Preskill, 2018].

Chapter 8

ZX-diagrams for variational quantum algorithms

Due to the relative simplicity of graphical manipulations comparing to matrix multiplications ZX-calculus is widely used to reason about applications from the quantum computing. Recently, ZX-calculus was applied to analyze *parameterized quantum circuits* [Zhao and Gao, 2021, Stollenwerk and Hadfield, 2022, Fontana *et al.*, 2020].

Parameterized quantum circuits (PQC) are essential building blocks of *variational quantum algorithms* (VQA). In the variational algorithms the parameters of the circuit are trained in a classical loop. The classical loop, in turn, may use the quantum computer to evaluate the *loss function*. The loss function is usually expressed as an expectation of some observable in a parameterized quantum state:

$$L(\boldsymbol{\theta}) = \langle 0 \dots 0 | U^\dagger(\boldsymbol{\theta}) O U(\boldsymbol{\theta}) | 0 \dots 0 \rangle \quad (8.1)$$

8.1 The diagram for QAOA circuit

In this work we concentrate on the *Quantum Approximate optimization Algorithm* (QAOA). We recall that QAOA operates the circuit:

$$|\psi_p(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = U_M(\beta_{m-1}) U_P(\gamma_{m-1}) \dots U_M(\beta_0) U_P(\gamma_0) |\psi_0\rangle \quad (8.2)$$

where $U_M(\beta) = e^{i\beta \sum_i X_i}$ and $U_P(\gamma) = e^{i\gamma C}$ for a diagonal operator C . We will see in this section that the parameterized unitary 8.2 can be decomposed in terms of $R_X(\beta)$, $R_Z(\gamma)$, H and CNOT gates and easily translated to a ZX-diagram.

We recall that in previously used notations the time in a ZX-diagram flows vertically from top to the bottom. In this chapter in order to keep the parallel with the circuit notation we will adopt the *horizontal time line*. Therefore, the input of the diagram is on the left and the output on the right. As most of the diagrams in this section are scalars, the particular time direction actually doesn't matter at all.

Phase gadgets

When QAOA is used to solve a combinatorial optimization problem the operator C encodes the objective function. For *constraint satisfaction* optimization problems the objective function

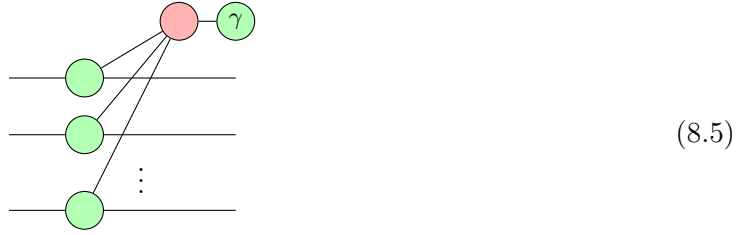
$f : \{0, 1\}^n \rightarrow \mathbb{R}$ is usually given as:

$$f(\mathbf{x}) = \sum_{S \subset [1, \dots, n]} c_S \bigoplus_{i \in S} x_i \quad (8.3)$$

For example, for the MaxCut problem we have:

$$f(\mathbf{x}) = \sum_{(u,v) \in E} e_{u,v} x_u \oplus x_v \quad (8.4)$$

A unitary $U_{\mathcal{X}}(\gamma) = e^{i\gamma x_{i_1} \oplus \dots \oplus x_{i_l}}$ acting on qubits from the set \mathcal{X} corresponds (up to a scalar λ) to a *phase gadget*. Phase gadgets are ZX-diagrams of the form:



up to a scalar factor λ .

The work [Cowtan *et al.*, 2020] is dedicated to the properties of phase gadgets and their generalization called *phase polynomials*. We provide a small example showing how obtain a phase polynomial diagram representing $e^{i\gamma C}$ for an Ising Hamiltonian C :

$$C = \sum_{i=1}^n h_i Z_i + \sum_{1 \leq i < j \leq n} h_{i,j} Z_i Z_j \quad (8.6)$$

A ZX-diagram for the term $e^{i\gamma C}$ can be composed out of diagrams for individual and pair terms $e^{i\gamma h_i Z_i}$ and $e^{i\gamma h_{i,j} Z_i Z_j}$. For the one-qubit term $e^{i\gamma h_i Z_i}$ the diagram is simply $\text{---} \text{green circle with } 2h_i\gamma \text{---}$ up to the phase scalar $e^{-i\gamma h_i}$.

For the pair interaction we know that

$$e^{i\gamma Z_i Z_j} = e^{i\gamma} \text{CNOT } R_Z(-2\gamma) \text{CNOT} \quad (8.7)$$

Up to a scalar that doesn't depend on γ , the diagrams for the number $e^{i\gamma}$ and for the gate CNOT are and respectively. Therefore, we can obtain a diagram for $e^{i\gamma Z_i Z_j}$ using the composition:

(8.8)

We introduce the shortcuts $\gamma_i = 2\gamma h_i$ and $\gamma_{i,j} = 2\gamma h_{i,j}$. Using phase gadgets we can easily draw a ZX-diagram for the parameterized $e^{i\gamma C}$:

(8.9)

In this diagram we dropped the global complex phase as it is not meaningful in our context.

The uniform superposition $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = |+\rangle \otimes \cdots \otimes |+\rangle$ that is used as the initial state can be represented by the diagram $n \left\{ \begin{array}{c} \text{green circle} \\ \vdots \\ \text{green circle} \end{array} \right.$. The diagram for the mixing operator $U_M(\beta) = \prod_{1 \leq i \leq n} e^{i\beta X_i}$ is also straightforward: $n \left\{ \begin{array}{c} \text{red circle labeled } \beta \\ \vdots \\ \text{red circle labeled } \beta \end{array} \right.$

By combining the building blocks we obtain a diagram for the one-depth QAOA state $|\psi(\beta, \gamma)\rangle = e^{i\beta \sum_i X_i} e^{i\gamma C} |+\rangle + \cdots +$:

(8.10)

8.2 Loss function

The classical loop of QAOA optimizes the *loss function*. The loss function is given by the expectation of the operator C in the quantum state:

$$\langle \psi(\beta, \gamma) | C | \psi(\beta, \gamma) \rangle \quad (8.11)$$

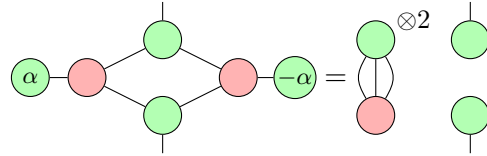
For the Ising Hamiltonian the energy expectation can be computed as the weighted sum of individual contributions $\langle Z_v \rangle_{\beta, \gamma}$ and $\langle Z_u Z_v \rangle_{\beta, \gamma}$:

$$\langle C \rangle_{\beta, \gamma} = \sum_{i=1}^n h_i \langle Z_i \rangle_{\beta, \gamma} + \sum_{1 \leq i < j \leq n} h_{i,j} \langle Z_i Z_j \rangle_{\beta, \gamma} \quad (8.12)$$

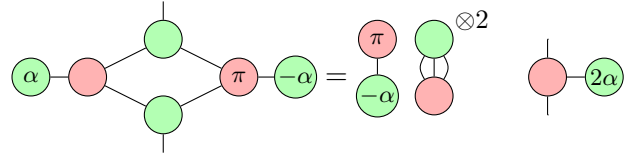
In the next chapter we will see how to represent such sum as a ZX-diagram. For now, we will consider only the individual terms $\langle Z_v \rangle_{\beta, \gamma}$ and $\langle Z_u Z_v \rangle_{\beta, \gamma}$. In particular, we will show how to derive the analytical expression for these values with the means of ZX-calculus.

In this section we will demonstrate how to use ZX-calculus to derive an analytical formula for *general Ising models*. This result was derived as a part of the thesis project and is presented in the current manuscript for the first time. We remark that a similar objective was pursued in [Stollenwerk and Hadfield, 2022] where ZX-diagrams are used to re-derive the formula for *unweighted MaxCut*. Contrary to [Stollenwerk and Hadfield, 2022], we don't proceed by an explicit extension of the traditional framework. Instead, we locally use a decomposition of the identity matrix.

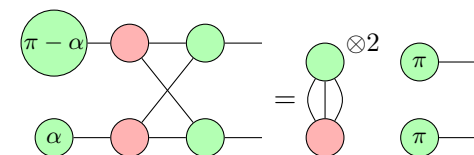
Lemma 8.2.4



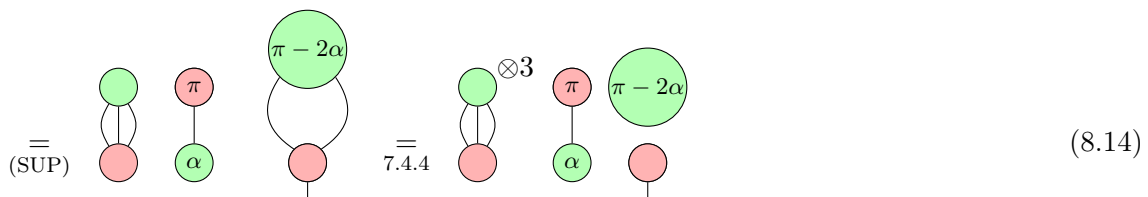
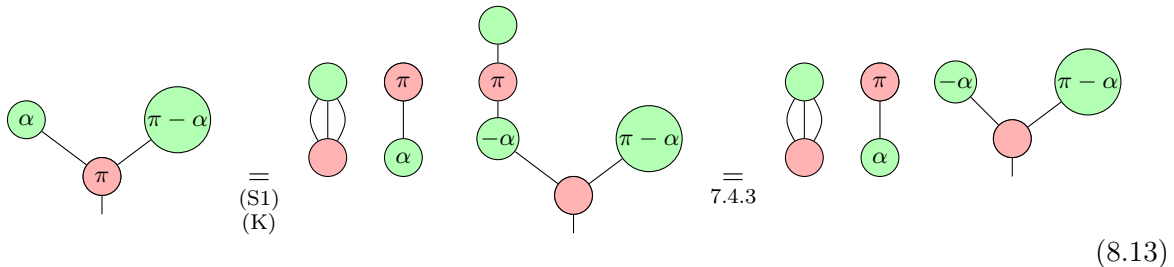
Lemma 8.2.5



Lemma 8.2.6



Proof: 8.2.2

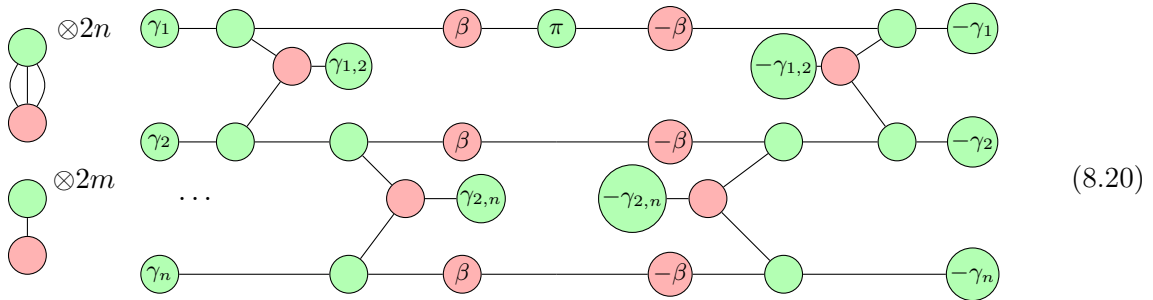


8.2.1 The mean value of $\langle Z_i \rangle$

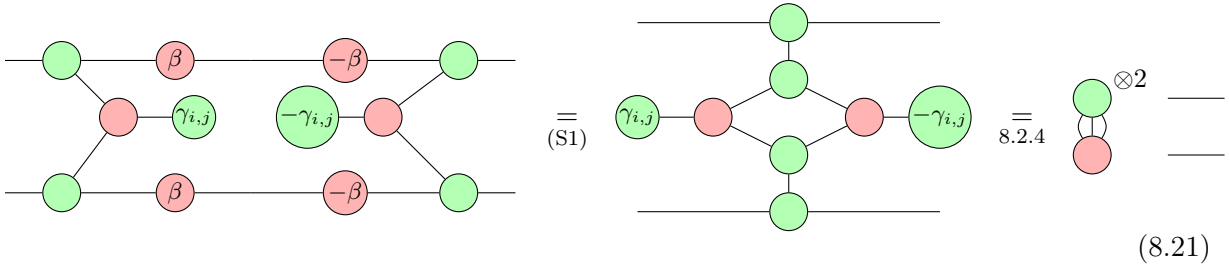
The diagram for the operator Z is simply $-\pi$.

We can evaluate the expectation $\langle \psi(\beta, \gamma) | Z | \psi(\beta, \gamma) \rangle$ by explicitly composing diagrams for the operator Z , the state $|\psi(\beta, \gamma)\rangle$ and its adjoint. We recall that the diagram for the adjoint $\langle \psi(\beta, \gamma) |$ is precisely the diagram for $|\psi(-\beta, -\gamma)\rangle$ flipped from left to right.

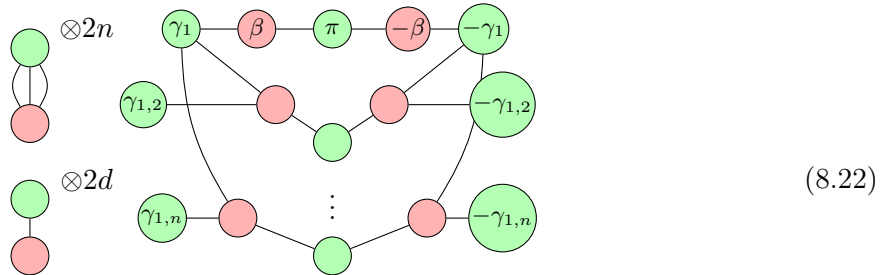
The diagram for $\langle \psi(\beta, \gamma) | Z | \psi(\beta, \gamma) \rangle$ is:



We remark that all phase gadgets connecting qubits i and j that are not acted by the Pauli operator cancel while leaving a constant factor:

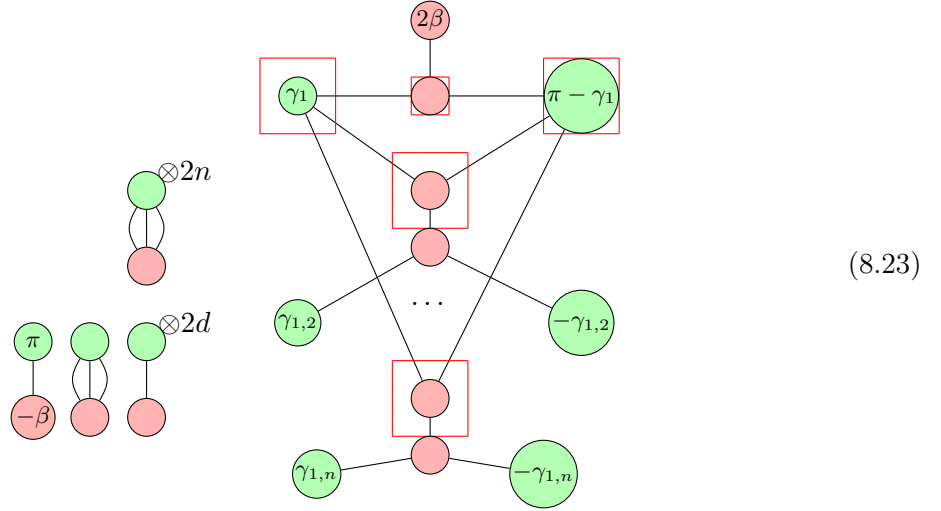


Such simplifications lead to the diagram:



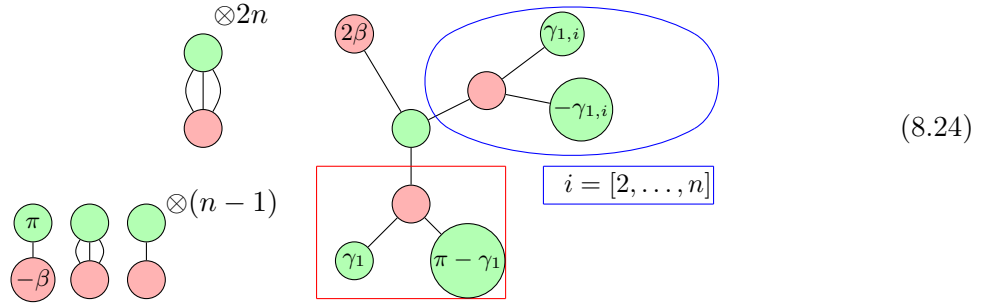
We use (K) to move the π -spider to the left and (S1) to fuse and unfuse spiders of the same

colors. We obtain the diagram:

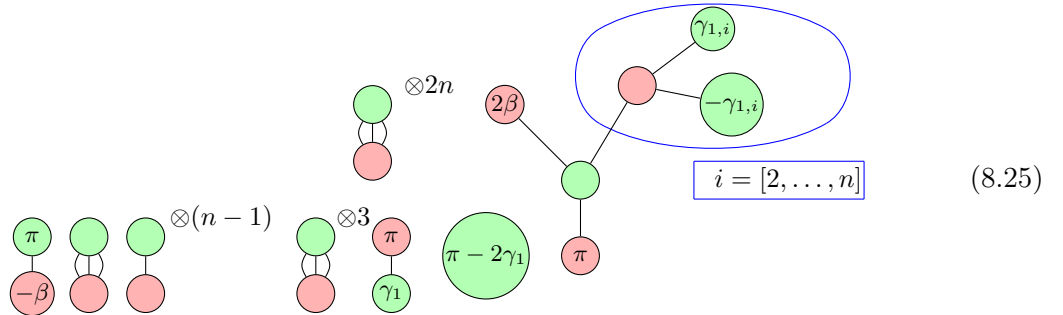


where d is the degree of the node 1. We assume that the connectivity graph of the Hamiltonian C is full. A missing edge is simply an edge of zero weight. Therefore, in our settings $d = n - 1$.

We can apply the generalized bialgebra rule (7.4.5) to spiders. After the rule application, the diagram becomes:



Then we apply the lemma 8.2.3 to obtain:



Finally, the lemmas 7.4.2 and 8.2.1 lead us to a diagram that is a simple multiplication of

known cosines and sinus scalars (8.18):

$$(8.26)$$

$$(8.27)$$

Using the result 8.18, we can directly compute the scalar value of the final diagram. This value is:

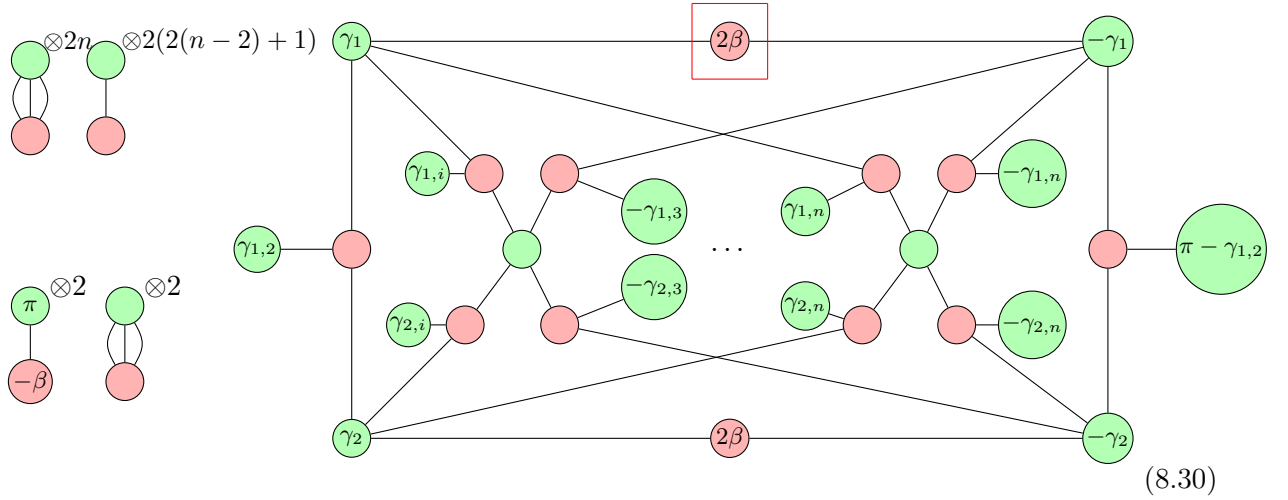
$$\langle \psi(\beta, \gamma) | Z | \psi(\beta, \gamma) \rangle \stackrel{8.18}{=} \sin \beta \sin \gamma_1 \prod_{i=2}^n \cos \gamma_{1,i} \quad (8.28)$$

8.2.2 The mean value of $\langle Z_i Z_j \rangle$

In the previous section we managed to compute $\langle \psi(\beta, \gamma) | Z | \psi(\beta, \gamma) \rangle$ using only graphical rewritings for internal transformations. For the two-qubit operator $Z_i Z_j$ the situation is more complicated. Firstly, we start with the diagram:

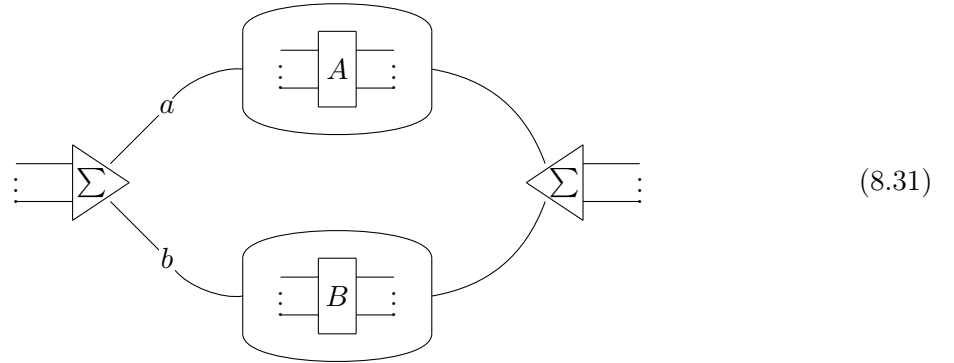
$$(8.29)$$

By performing spider fusion and commuting the π -spider with (K) rule we get a slightly easier representation.



At this step it is unclear how to proceed further. Indeed, linear rules don't seem to simplify much the diagram. Moreover, we recall that the diagram equals to a complex formula for the mean value in QAOA state (3.37). The formula contains sums and can't be decomposed in a product of cosines and sinuses with angles that are linear on parameters β and γ . Therefore, the diagram (8.30) is probably the simplest one that express the desired mean value.

At this step the work [Stollenwerk and Hadfield, 2022] suggests to explicitly decompose Pauli rotations. The decomposition is integrated in the language with a new sum-gadget that represents a linear combination of diagrams:



where a and b are the coefficients of the decomposition. The diagrams $\vdash A \vdash$ and $\vdash B \vdash$ have the same number of input and output wires. The new gadget comes with *product pull rule* and *diagram pull rule*. The *product pull rule* allows to compose two subsequent sum-gadgets. The *diagram pull rule* essentially embodies the distributivity of addition with respect to tensor product and matrix multiplication.

In [Stollenwerk and Hadfield, 2022] the sum-gadget is used to commute Z -spiders and X spiders in non-trivial cases when angles are not 0 or π . For the mean values appearing in QAOA the work extensively uses the decomposition of the spiders corresponding to Pauli rotations:

$$e^{i\gamma} \text{---} \text{---} (-2\gamma) \text{---} = \text{---} \left[\begin{array}{c} \cos \gamma \\ i \sin \gamma \end{array} \right] \left[\begin{array}{c} \text{---} \\ \text{---} \pi \text{---} \end{array} \right] \text{---} \Sigma \text{---} \quad (8.32)$$

We remark that explicit linear combinations of diagrams, alternatively called *bags of diagrams* were already used in [Toumi *et al.*, 2021] in the context of differentiation. In fact, until the recent paper [Yeung *et al.*,] and our contribution [Jeandel *et al.*, 2022] it was the only way to deal with sums. In the next chapter we will present how to add diagrams entirely diagrammatically.

To derive the analytical formula, we suggest to use a more restrictive set of tools than the one presented in [Stollenwerk and Hadfield, 2022]. Actually, the only thing we need is the trivial decomposition of the identity matrix:

$$\llbracket \text{---} \rrbracket = \frac{1}{2} \left(\llbracket \text{---} \text{---} \rrbracket + \llbracket \text{---} \pi \text{---} \pi \text{---} \rrbracket \right) \quad (8.33)$$

For linear maps of same dimensions the addition can be naturally defined. Trivially, a sum of two matrices can be obtained with entry-wise addition of their elements. A formal definition for sum operator is given in [Coecke and Kissinger, 2017] (chapter 5). According to this definition, the sum operator is distributive with respect to tensor product and composition. The distributive property implies that we can apply the decomposition (8.33) locally for an internal wire of the diagram:

$$\llbracket \text{---} \text{---} \rrbracket = \frac{1}{2} \left(\llbracket \text{---} \text{---} \text{---} \rrbracket + \llbracket \text{---} \pi \text{---} \pi \text{---} \rrbracket \right) \quad (8.34)$$

Using this decomposition, we can prove the following lemma:

Lemma 8.2.7

$$\left[\text{Diagram} \right] \quad (8.35)$$

$$= \cos(\alpha_0 + \beta_0) \prod_{i=1}^s \cos(\alpha_i + \beta_i) + \cos(\alpha_0 - \beta_0) \prod_{i=1}^s \cos(\alpha_i - \beta_i) \quad (8.36)$$

Proof:

$$\left[\begin{array}{c} \text{Diagram} \end{array} \right] = \frac{1}{2} (\llbracket A \rrbracket + \llbracket B \rrbracket) \quad (8.37)$$

The diagram inside the brackets shows a sequence of nodes: a green node with two self-loops labeled $\otimes 2$, a red node labeled π , a green circle labeled $-\alpha_0 - \beta_0$, a green node labeled $2\alpha_0$, a red node labeled $2\beta_0$, a red rectangle, a green node, a red node, a green node labeled $2\alpha_i$, a red node labeled $2\beta_i$, a green circle labeled $-\alpha_i - \beta_i$, a red node labeled π , and a green node with two self-loops labeled $\otimes 2$. A blue oval encloses the nodes from $2\alpha_i$ to $2\beta_i$. A blue box below the oval is labeled $i = [1, \dots, s]$.

We apply the decomposition to the wire surrounded by red rectangle. The first term becomes:

$$\begin{aligned} A &= \text{Diagram} \\ &\stackrel{(B1)}{=} \text{Diagram} \\ &\stackrel{8.18}{\longrightarrow} \boxed{2 \cos(\alpha_0 + \beta_0) \prod_{i=1}^s \cos(\alpha_i + \beta_i)} \end{aligned} \quad (8.38)$$

The first diagram shows the decomposition of the red rectangle into two red rectangles. The second diagram shows the result of applying rule (B1), where the red rectangle is now a green node. The third diagram shows the result of applying rule 8.18, which is a boxed expression.

The second term may be computed similarly:

$$B = \text{Diagram}$$

The diagram shows a sequence of nodes: a green node with two self-loops labeled $\otimes 2$, a red node labeled π , a green circle labeled $-\alpha_0 - \beta_0$, a green node labeled $2\alpha_0$, a red node labeled π , a green node labeled $2\beta_0$, a red rectangle, a green node, a red node, a green node labeled $2\alpha_i$, a red node labeled $2\beta_i$, a green circle labeled $-\alpha_i - \beta_i$, a red node labeled π , and a green node with two self-loops labeled $\otimes 2$. A blue oval encloses the nodes from $2\alpha_i$ to $2\beta_i$. A blue box below the oval is labeled $i = [1, \dots, s]$.

$$\begin{aligned}
& \stackrel{(B1)}{=} \text{Diagram 1} \\
& \stackrel{8.2.1}{=} \text{Diagram 2} \\
& \stackrel{7.4.7}{=} \stackrel{7.4.6}{=} \text{Diagram 3} \\
& \xrightarrow{8.18} \boxed{2 \cos(\alpha_0 - \beta_0) \prod_{i=1}^s \cos(\alpha_i - \beta_i)} \tag{8.39}
\end{aligned}$$

$i = [1, \dots, s]$

■

We will use the identity decomposition to compute the value of the diagram (8.30). In a manner, our graphical computations will be assisted with the semantics equalities.

Firstly, we transform the part of the diagram (8.30) in the red frame.

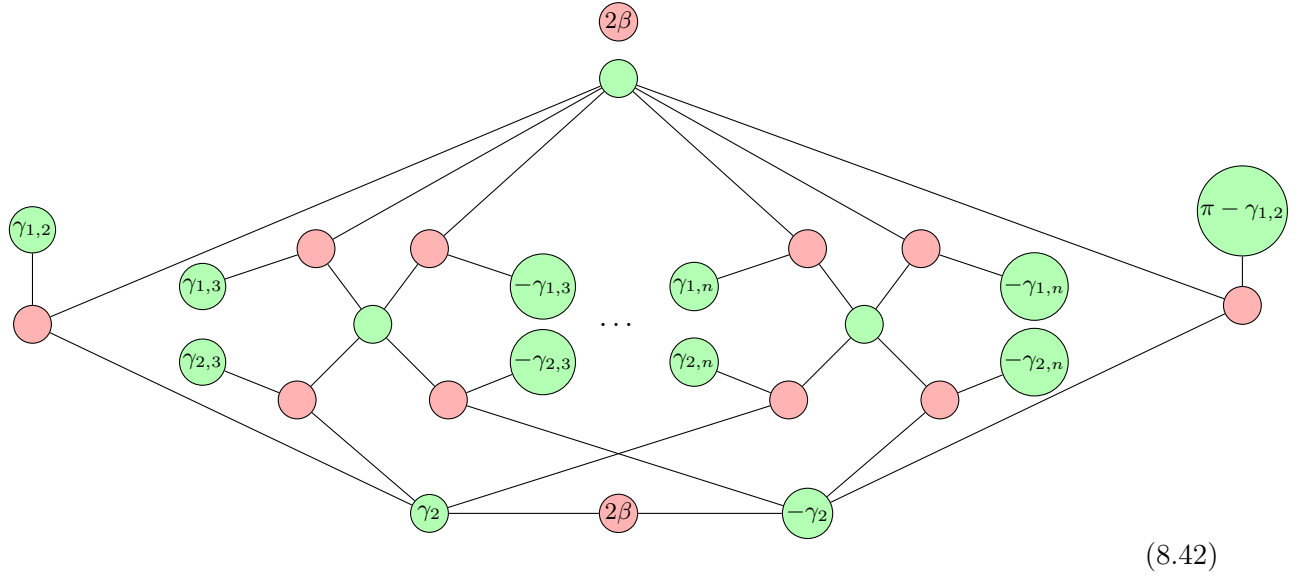
$$\left\| \begin{array}{c} \text{red circle with } 2\beta \\ \text{red circle with } \pi \end{array} \right\| = \frac{1}{2} \left(\left\| \underbrace{\begin{array}{c} \text{red circle with } 2\beta \\ \text{red circle with } \pi \end{array}}_A \right\| + \left\| \underbrace{\begin{array}{c} \text{red circle with } \pi + 2\beta \\ \text{red circle with } \pi \end{array}}_B \right\| \right) \tag{8.40}$$

We now reduce overall diagram for each part of the decomposition (8.40). For the moment, we will ignore the common scalar factor

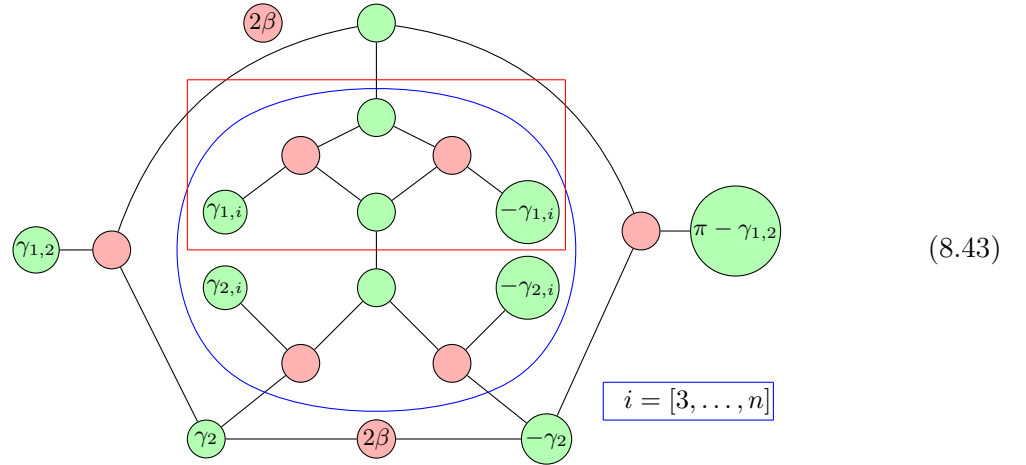
$$s = \frac{1}{2} \left(\text{Diagram 4} \right) \stackrel{7.4.6}{=} \stackrel{7.3.1}{=} \text{Diagram 5} \tag{8.41}$$

The scalar factor will be added in the final steps of the computation.

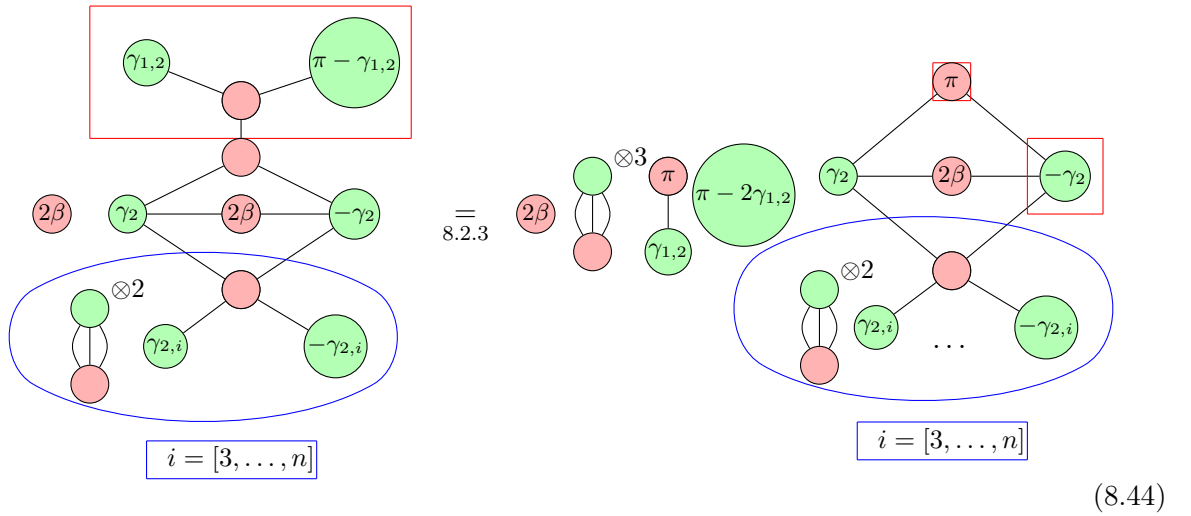
For the first part corresponding to A in (8.40) after the application of (S2) and (S1) we obtain:



With the blue frame the diagram can be written in a more compact way.



We apply the lemma 8.2.4 to the part in red frame.



With the lemma 7.4.8 that allows to push red π -spider through the green spider and the rule (S1) we get:

$$(8.45)$$

$$(8.46)$$

Multiplying this diagram by the scalar factor we obtain with the aid of lemmas 7.4.6 and 8.18 the formula:

$$\frac{1}{2} \llbracket A \rrbracket = \sin \beta \cos \beta \sin \gamma_{1,2} \cos \gamma_2 \prod_{i=3}^n \cos \gamma_{2,i} \quad (8.47)$$

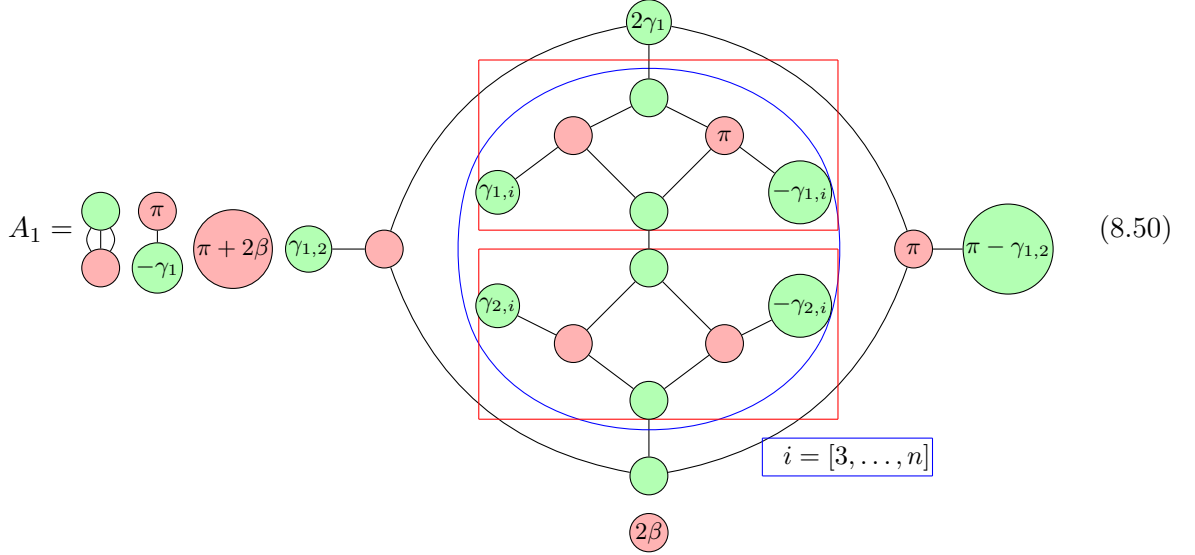
For the part that correspond to B in the decomposition we have:

$$(8.48)$$

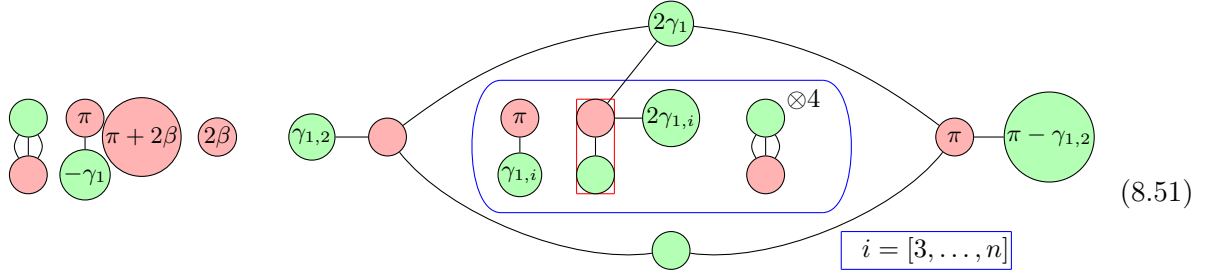
We apply one more time the decomposition (8.40) for the bottom spider. We obtain two diagrams A_1 and B_1 :

$$\llbracket B \rrbracket = \frac{1}{2} (\llbracket A_1 \rrbracket + \llbracket B_1 \rrbracket) \quad (8.49)$$

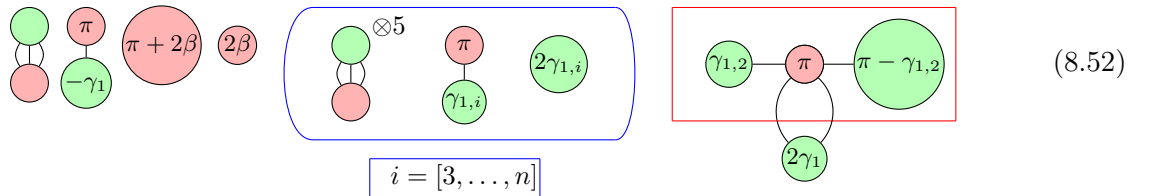
The first diagram A_1 is



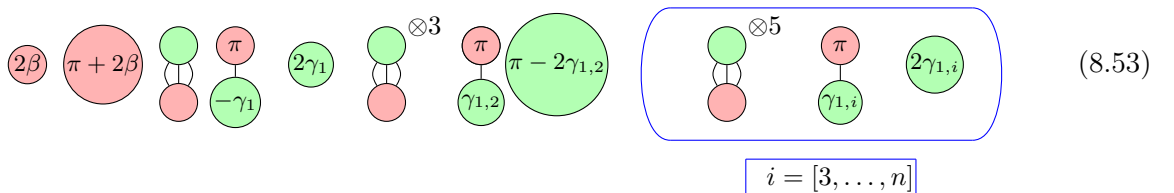
We simplify A_1 firstly with the lemmas 8.2.5 and 8.2.4:



Then the application of (S2) and (B1) leads to:



We use the lemma 8.2.3 and the rules (S1), (S2) to find the result:



We remark this diagram is almost the same as 8.46 with interchanged indexes 1 and 2. Therefore, by multiplying this diagram by the factor with $\frac{1}{2}$ resulting from the (8.49) we obtain an expression similar to (8.47):

$$\frac{1}{2} \llbracket sA_1 \rrbracket = \sin \beta \cos \beta \sin \gamma_{1,2} \cos \gamma_1 \prod_{i=3}^n \cos \gamma_{1,i} \quad (8.54)$$

The second term corresponding to B_1 is reduced as follows:

$$B_1 = \text{Diagram} \quad (8.55)$$

Firstly we apply the lemma 8.2.5.

$$\text{Diagram} \quad (8.56)$$

Then we simply apply the (S1) to reproduce the condition of the lemma 8.2.6:

$$\text{Diagram} \quad (8.57)$$

Applying 8.2.6 we obtain:

$$(8.58)$$

We combine scalar phases with 7.4.7:

$$(8.59)$$

At the last step we use the generalized bialgebra rule 7.4.5:

$$(8.60)$$

We've already seen a similar diagram in the lemma 8.2.7. After the multiplication by the scalar $s' = s/2$ we get the last component of the mean value formula:

$$\begin{aligned} \frac{1}{2} \llbracket sB_1 \rrbracket &= -\sin^2 \beta (-1) \left(\cos(\gamma_1 + \gamma_2 + \pi) \prod_{i=3}^n \cos(\gamma_{1,i} + \gamma_{2,i}) + \cos(\gamma_1 - \gamma_2) \prod_{i=3}^n \cos(\gamma_{1,i} - \gamma_{2,i}) \right) \\ &= \sin^2 \beta \left(\cos(\gamma_1 - \gamma_2) \prod_{i=3}^n \cos(\gamma_{1,i} - \gamma_{2,i}) - \cos(\gamma_1 + \gamma_2) \prod_{i=3}^n \cos(\gamma_{1,i} + \gamma_{2,i}) \right) \end{aligned} \quad (8.61)$$

Combining the terms (8.47), (8.54), (8.61) we obtain:

$$\begin{aligned} \langle \psi(\beta, \gamma) | Z_1 Z_2 | \psi(\beta, \gamma) \rangle = & \frac{\sin 2\beta}{2} \sin \gamma_{1,2} \left(\cos \gamma_1 \prod_{i=3}^n \cos \gamma_{1,i} + \cos \gamma_2 \prod_{i=3}^n \cos \gamma_{2,i} \right) + \\ & + \sin^2 \beta \left(\cos(\gamma_1 - \gamma_2) \prod_{i=3}^n \cos(\gamma_{1,i} - \gamma_{2,i}) - \cos(\gamma_1 + \gamma_2) \prod_{i=3}^n \cos(\gamma_{1,i} + \gamma_{2,i}) \right) \end{aligned} \quad (8.62)$$

8.3 Discussion

In the previous section we used the rules of ZX-calculus to derive formulas (8.28) and (8.62) for the loss function in QAOA of depth $p = 1$. Our result was derived for a *general Ising model*. Our analytical formula are extremely helpful for numerical simulations. For instance, the efficient classical simulation of RQAOA with depth $p = 1$ used to obtain the results presented in sections 5.3.3 and 6.4.3 is owed to this formula. In addition, our original demonstration performed in the framework of ZX-calculus allows to connect two important domains of the quantum computing: *ZX-calculus* and *variational algorithms*.

In this particular example we can hardly claim that the diagrammatic representation made the computation easier. Moreover, sometimes we had to use semantic equalities such that (8.33) to be able to continue the simplifications. Nevertheless, our computation pointed out several interesting results.

Firstly, we demonstrated how to express the routines of QAOA in the framework of ZX-calculus. Such diagrammatic representation can have its own merits. Possible application is the study of the parameter landscape. For instance, in the context of quantum machine learning ZX-diagrams were used to reason about the barren plateau phenomena [Zhao and Gao, 2021].

Secondly, we have shown how to perform the computation for a *difficult quantum algorithm* by manipulating the diagrams with the rewrite rules. We also suggested a combined approach that outsources only a part of computation to the graphical rewriting. This combination passed by the semantical decomposition of *identity* on a sum of two projectors 8.33. In [Stollenwerk and Hadfield, 2022] a slightly different technique was used. In this alternative technique the spiders corresponding to *Pauli gates* were decomposed in a linear combination.

Finally, we have observed that despite multiple advantages of the graphical representation, the framework was still lacking tools that are important to reason about variational algorithms. One such desirable tool is *addition*. Indeed, in the traditional ZX-calculus there is no an efficient way to extract a diagram from the expression corresponding the Ising Hamiltonian 8.6. Obviously, as ZX-diagrams are universal such diagram exists. Moreover, it can be constructed from the explicit matrix representation while the complexity of such procedure is exponential.

The other missing tool is *differentiation*. It is particularly important to reason about the training of the variational circuit. In the next chapter we will present our original technique for the graphical *addition* and *differentiation* of ZX-diagrams.

Chapter 9

Addition and Differentiation of ZX-diagrams

This chapter is based on our results presented in [Jeandel *et al.*, 2022].

9.1 Motivation

Due to its flexibility, ZX-calculus is widely used to address different problems of quantum computing. However, its application to the rapidly growing field of variational algorithms [Cerezo *et al.*, 2021a] such as QAOA [Farhi *et al.*, 2014a], VQE [Peruzzo *et al.*, 2014] and variational quantum machine learning are so far limited. Nevertheless as variational algorithms do not require heavy resource for error-correction, the incoming emergence of *NISQ* devices makes from them an object of particular attention [Preskill, 2018].

In the previous chapter we have shown how to derive the analytical expression for the loss function in a particular case of QAOA with depth $p = 1$. We remark that at some point we had to decompose the diagram on a sum of two terms. The resulting computations were quite tedious. The advantage of diagrammatic representation was insignificant.

However, we believe that the study of variational algorithms with the means of ZX-calculus is a promising research direction. For instance, the diagrammatic representation is beneficial in the analysis of the barren plateau phenomena [Zhao and Gao, 2021]. Yet the work [Zhao and Gao, 2021] pointed out a crucial obstacle limiting the application of ZX-calculus to variational algorithms. This obstacle turns out to be the absence of convenient tools for *addition* and *differentiation* of parametrized diagrams.


Indeed, basic building blocks of variational algorithms are parametrized circuits and the search of optimal parameter values is a crucial part of these algorithms. The search is usually done by classical numerical optimization methods and most of them use derivatives [Guerreschi and Smelyanskiy, 2019].

Addition is closely related to *differentiation* via *product rules*. *Product rules* are crucial to evaluate the derivative of the composition and the tensor product.

Furthermore, in variational algorithms the loss function is typically given as the expectation of a local Hamiltonian in a parameterized quantum state. Addition of diagrams may be used to obtain graphical representations of such *Hamiltonians*. We recall that Hamiltonians are non-unitary Hermitian operators corresponding to the energy of the system. In practical applications

Hamiltonians are usually given as a sum of local terms:

$$H = \sum_{i=1}^k H_i \quad (9.1)$$

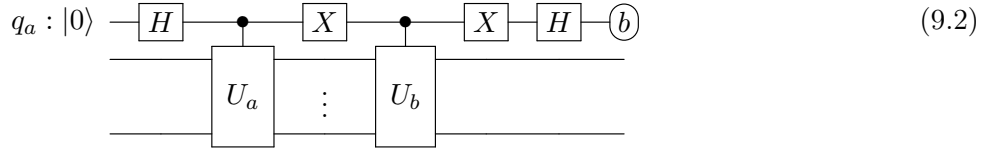
where each H_i acts only on a limited number of qubits. Usually, individual terms H_i can be relatively easily represented as diagrams. For instance, for the Ising Hamiltonian $H_i = Z_u Z_v$ and the diagram for individual terms is trivial: . However, in the traditional framework there is no graphical procedure that constructs the overall sum out of such elementary blocks.

9.1.1 Addition in quantum computing

In pure qubit quantum mechanics computations are performed by unitary transformations. Unitary transformations form a group with respect to multiplication. On the other side, a sum of two unitary transformations U_a and U_b is not unitary. However, in some context it is meaningful to consider transformations of the form $U_a + U_b$.

For instance, the simulation of an evolution under some Hamiltonian may be done more efficiently using linear combination of unitary operators [Childs and Wiebe, 2012]. The Hamiltonian simulation problem that is central problem for quantum chemistry. The traditional solution for the problem uses Lie-Trotter-Suzuki *product formulas*. As was shown in [Childs and Wiebe, 2012] this standard approach may outperformed by using linear combination of unitaries.

The work [Childs and Wiebe, 2012] suggests the following protocol for the addition of unitaries:



where q_a is an ancilla qubit initialized in a zero-state, and unitary gates U_a and U_b are conditioned on an ancilla.

The protocol (9.2) can be easily verified:

$$|0\rangle|\psi\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}}|\psi\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle \otimes U_b|\psi\rangle + |1\rangle \otimes U_a|\psi\rangle) \quad (9.3)$$

$$\rightarrow \frac{1}{2}(|0\rangle \otimes (U_b + U_a)|\psi\rangle + |1\rangle \otimes (U_b - U_a)|\psi\rangle) \quad (9.4)$$

If the ancilla qubit is measured in $|0\rangle$ state, the input state $|\psi\rangle$ was successfully transformed to $\frac{1}{c}(U_a + U_b)|\psi\rangle$ where c is the normalization constant.

We remark that the protocol (9.2) uses an ancilla qubit and post-selection on the measurement output. Therefore, it exceeds the framework of pure qubit computations. We also emphasize that the circuit involves *controlled versions* of unitary transformations. In order to be executed on a quantum computer, the controlled unitaries have to be compiled to a sequence of elementary gates. An efficient compilation is non-trivial even for a basic controlled-CNOT gate usually called *Toffoli gate*. Interestingly, a graphical representation of controlled gates is particularly natural in ZH-calculus [Backens and Kissinger, 2019]. The graphical representation may enhance the compilation of controlled gates [Kuijpers *et al.*, 2019].

Traditional ZX-calculus is not equipped with a protocol allowing the linear combinations of arbitrary ZX-diagrams. Fundamentally, this deficiency is due to the absence of a proper physical interpretation for a sum of diagrams. Indeed, ZX-calculus is a language for process theory representation of quantum computing [Coecke and Kissinger, 2017]. In other words, each diagram is a process. The composition of processes can be interpreted as their sequential application. The tensor product corresponds to the parallel application. In the context of process theory, the addition doesn't have a meaningful physical interpretation.

Several attempts were made to address this issue. The first option selected in [Toumi *et al.*, 2021, Stollenwerk and Hadfield, 2022] consists in an extension of the formalism from diagrams to *bags of diagrams*. More precisely, this approach suggest to consider formal sums of diagrams as an object of study. Unfortunately, there is no rules to manipulate sums of ZX-diagrams. Therefore, the interest of graphical computations is significantly reduced. In the previous chapter we have seen an example application of this approach confirming our doubts.

In [Jeandel *et al.*, 2022] we have introduced an original technique that allows to add diagrams entirely graphically. In our approach a sum of diagrams is another diagram. Interestingly, we derived the rule for diagrammatic addition using controlled versions of diagrams. We suggest an inductive procedure that allows to construct a controlled version for arbitrary diagram. Our result with be presented in the section 9.2.

An independent work [Yeung *et al.*,] suggests an alternative protocol for diagrammatic addition for slightly more expressive algebraic ZXW-calculus. This result is also based on a translation to a controlled form. However, their definition of a controlled form is not equivalent to ours. In addition, instead of inductive procedure, they rely on a decomposition of a diagram on elementary matrices that have an exponential complexity in the worst-case. Later in this chapter we will provide a more detailed comparison of two approaches.

Representing Hamiltonians as ZX-diagram

The addition of diagrams naturally appears in the representation of Hamiltonians. Usually, Hamiltonians are given as a weighted sum of Pauli tensors. Pauli tensors have a particularly simple decomposition on elementary matrices. Therefore, the procedure described in [Yeung *et al.*,] is well-suited for the construction of diagrammatic representation for such Hamiltonians.

In addition, the work [Yeung *et al.*,] also suggests a way to *exponentiate* diagrams representing Hamiltonians. The exponentiate of a Hamiltonian H is a unitary $e^{i\theta H}$. The exponentiation is done with the aid of the *Cayley-Hamilton theorem*. The Cayley-Hamilton theorem allows to decompose the exponent $e^{i\theta H}$ on a linear combination of $H^k = \underbrace{H \times \cdots \times H}_k$ for $k \in [1, \dots, 2^n]$.

We remark that in a particular case when $H = \sum_i H_i$ is a sum of commuting terms it is usually straightforward to obtain $e^{i\theta H}$ as a product of individual $e^{i\theta H_i}$. However, if some local terms in H don't commute the product decomposition doesn't apply. For instance, this is the case for *modified mixer Hamiltonians* used in the extension of QAOA to constrained optimization problems [Hadfield *et al.*, 2019].

In our work [Jeandel *et al.*, 2022] we considered an inverse problem. Using the *Stone's theorem*, we have demonstrated how to obtain a diagram for H given the *derivative* of the diagram for $e^{i\theta H}$. We illustrate this procedure on an example of Ising Hamiltonian. Our result in presented the Section 9.4.

The Stone's theorem provides a link between differentiation and addition. With this theorem some specific kind of sums may be straightforwardly expressed as diagrams if we can compute derivatives. On the other directions, the ability to perform addition directly leads to a procedure

for differentiation. Indeed, the major obstruction for diagrammatic differentiation comes from the *product rules* that involve sums. In the Section 9.3.1 we will show how an inductive definition of the derivative is obtained by explicit diagrammatic representation of the product rules.

9.1.2 Differentiation in variational algorithms

In the previous chapter we have seen that the *loss function* in variational algorithms can be represented as a parameterized scalar ZX-diagram. Many state-of-the art methods for the optimization of the loss function such as *Quantum Analytic Descent* [Koczor and Benjamin, 2022] and *meta-learning optimizers* [Wilson *et al.*, 2021] use derivatives.

For an efficient training it is important to detect such deficiencies in the parameter landscape as *barren plateau*. In the context of variational algorithms, it was demonstrated that a specific structure of the *cost function* may lead to such phenomena [Cerezo *et al.*, 2021b]. Alternatively, the structure of the *ansatz itself* may lead to the exponentially vanishing gradient [McClean *et al.*, 2018].

For the first time a graphical study of barren plateau was presented in [Zhao and Gao, 2021]. This work analyzes the diagrams with only two occurrences of the parameter. In [Toumi *et al.*, 2021] an arbitrary number of parameter occurrences is tackled with explicitly written product rules leading to *bags of diagrams*.

Alternatively, diagrammatic representation can be used to design *parameter shift rules*.

Parameter shift rules

Parameter shift rules appeared as a solution to the problem of gradient evaluation. Indeed, as we have pointed out in the Section 3.5.7 the gradient of the loss function $\frac{\partial \langle \psi_0 | U^\dagger(\alpha) O U(\alpha) | \psi_0 \rangle}{\partial \alpha_i}$ in general can't be directly evaluated on a quantum hardware. However sometimes we can express the gradient as a linear combination of the loss function values in different points:

$$\frac{\partial L}{\partial \alpha_{i^*}}(\hat{\alpha}) = \sum_{k=1}^m \epsilon_k L(\alpha^k) \quad (9.5)$$

where $\alpha_i^k = \begin{cases} \hat{\alpha}_i, & \forall i \neq i^* \\ \hat{\alpha}_i + \phi_i, & i = i^* \end{cases}$ The values $L(\alpha^k)$ can be computed with stochastic approximation.

Decompositions of the form (9.5) are called *parameter shift rules*.

In most cases such rules consider the parameterized operators of the form $U(\alpha) = e^{i\alpha H}$ for a Hermitian operator H called *generator*. For instance, the first shift rule was derived for the generators H with two eigenvalues $+1$ and -1 [Mitarai *et al.*, 2018]. It was further extended to generators with arbitrary symmetric eigenvalues $+r$ and $-r$ [Schuld *et al.*, 2019]:

$$\frac{\partial L}{\partial \alpha}(\alpha) = \frac{r}{4} \left(L\left(\alpha + \frac{\pi}{4r}\right) - L\left(\alpha - \frac{\pi}{4r}\right) \right) \quad (9.6)$$

An arbitrary parameterized transformation can be decomposed on a sequence of elementary gates $U(\alpha) = E_1(\alpha) \times \dots \times E_M(\alpha)$ where some of the gates depend on the parameter α . Typically, parameterized elementary gates are one-qubit Pauli rotations with two eigenvalues. Using the decomposition on elementary gates we can express gradients for small gates $E_i(\alpha)$ with the

parameter shift rule. The gradient of the overall unitary $U(\alpha)$ is computed with the product rule [Crooks, 2019]:

$$\frac{\partial U(\alpha)}{\partial \alpha} = \frac{\partial E_1(\alpha)}{\partial \alpha} \times \cdots \times E_M(\alpha) + \cdots + E_1(\alpha) \times \cdots \times \frac{\partial E_M(\alpha)}{\partial \alpha} \quad (9.7)$$

In the decomposition approach the number of shifts that require evaluation scales polynomially with the number of parameter occurrences. Alternatively, a *spectral decomposition* of H can be used to design a rule with reduced number of shifts. This approach was developed in [Wierichs *et al.*, 2022, Kyriienko and Elfving, 2021]. Two different algebraic approaches were presented in [Izmaylov *et al.*, 2021].

Such generalized rules are a priori more economical than the ones resulting from the decomposition on elementary gates. More precisely, the generalized rules lead to a smaller number of elements in the linear combination (9.5). As each value $L(\alpha^k)$ is typically evaluated on a quantum computer the number of terms in the linear combination is an important cost criteria. Remarkably, for the second-order derivatives the potential savings of generalized rules compared to decomposition approach are of order $O(n)$ [Wierichs *et al.*, 2022].

We remark that the parameter shift rules lead to an unbiased estimator for the gradient compared to the *finite difference approximation* $\frac{\partial L}{\partial \alpha}(\alpha_0) = \frac{L(\alpha_0 + \delta) - L(\alpha_0)}{\delta}$. Both approaches were compared analytically and numerically in [Mari *et al.*, 2021]. It was observed that parameter shift rules lead to more correct gradient estimations which, in turn, enhance the parameter optimization process.

We remark that even if we have a diagram for the derivative, we need to further extend the toolbox of ZX-calculus to suggest new parameter-shift rules. For instance, we may need a procedure that compiles a diagram into a linear combination of other diagrams corresponding to valid quantum expectations. In the diagrammatic analysis we don't have to systematically recover the structure of the initial circuit. As a consequence, we may end up with rules in a more general form:

$$\frac{\partial L}{\partial \alpha}(\alpha) = \langle \psi_1(\alpha) | O_1 | \psi_1(\alpha) \rangle + \cdots + \langle \psi_M(\alpha) | O_1 | \psi_M(\alpha) \rangle \quad (9.8)$$

Diagrammatic differentiation

The works [Zhao and Gao, 2021, Toumi *et al.*, 2021] use explicit sums of diagrams in order to represent the derivative of diagrams with multiple occurrences of the parameter. In contrast, in our approach the derivative of a parametrized ZX-diagram is another ZX-diagram. Hence we avoid the extension of the signature with formal sums. In order to tackle sums we use our original technique for the diagrammatic addition. An inductive definition of the derivative is obtained by explicit diagrammatic representation of the product rules. This inductive definition is presented in the Section 9.3.1.

In addition, in an attempt to give a ready-to-use toolbox for differentiation, we provide easy and convenient formulas to compute the derivative for the family of *linear diagrams* $ZX(\beta)$. These formulas are non-inductive but extremely fruitful for practical manipulations. Most of circuits for variational algorithms belong to $ZX(\beta)$ and we believe that our formulas will make the analysis of them much simpler.

A similar result was obtained in an independent work [Wang and Yeung, 2022]. This work uses W-spiders to handle product rules. In contrast to our result, the diagrammatic differentiation presented in [Wang and Yeung, 2022] associates to a ZX-diagram a diagram from another language called *algebraic ZX-calculus* [Wang, 2020]. The algebraic ZX-calculus is convenient to

represent arbitrary complex numbers, therefore their differentiation procedure can handle more general families of diagrams than $\text{ZX}(\beta)$. This advantage comes with the cost of abandoning the legacy of the vanilla ZX-calculus which is by far the most popular graphical calculus for quantum computing.

9.2 Addition of ZX-diagrams

The addition is a natural operation on matrices from a Hilbert space. In quantum mechanics it can be interpreted as the superposition phenomenon. However, the addition is not a physical process, hence it is not reflected in the standard ZX-calculus [Coecke and Kissinger, 2017].

On the other hand, for any two diagrams $D_1, D_2 : n \rightarrow m$, the universality of the ZX-calculus guarantees that there exists a diagram $D : n \rightarrow m$ such that $\llbracket D \rrbracket = \llbracket D_1 \rrbracket + \llbracket D_2 \rrbracket$. We provide in this section a general construction for such a diagram.

As pointed out in [Jeandel *et al.*, 2019] for the definition of normal forms in the ZX-calculus, one can inductively define the addition on 'controlled' versions of the diagrams. A controlled version of a diagram D_0 is roughly speaking a diagram with an extra input such that when this extra input is set to $|1\rangle$ the diagram behaves as D_0 and when it is set to $|0\rangle$ the diagram behaves as a neutral diagram. To build a controlled version of a diagram our construction pass by *controlled states* originally introduced in [Jeandel *et al.*, 2019]. In a nutshell, our procedure for addition consists in a way to represent every ZX-diagram by such a state.

9.2.1 Controlled states

Definition 9.2.1 (Controlled state [Jeandel *et al.*, 2019]) *A ZX-diagram $C : 1 \rightarrow n$ is a controlled state if $\llbracket C \rrbracket |0\rangle = \sum_{x \in \{0,1\}^n} |x\rangle = \left[\underbrace{\text{green circle} \dots \text{green circle}}_n \right]$.*

Intuitively, a controlled state is a way to encode the state $\llbracket C \rrbracket |1\rangle$. A controlled state with no outputs is called *controlled scalar*. For instance, the scalar 0 is encoded by a controlled scalar $C_0 = \text{green circle} \mid \text{red circle}$. Indeed, we can explicitly check that $\llbracket C_0 \rrbracket |0\rangle$ is a uniform superposition:

$$\left[\begin{array}{c} \text{green circle} \otimes 2 \\ \text{red circle} \end{array} \right] \stackrel{7.4.6}{=} \llbracket \text{red circle} \rrbracket = 1 \quad (9.9)$$

Verifying $\llbracket C_0 \rrbracket |1\rangle = 0$ completes the demonstration:

$$\left[\begin{array}{c} \text{green circle} \otimes 2 \\ \text{red circle} \end{array} \right] = \left[\begin{array}{c} \text{red circle} \otimes 2 \\ \text{green circle} \end{array} \right] \times \llbracket \pi \rrbracket = \left[\begin{array}{c} \text{red circle} \otimes 2 \\ \text{green circle} \end{array} \right] \times (1 + e^{i\pi}) = 0 \quad (9.10)$$

We remark that it is practical to use *graphical proofs* instead of matrix computation to manipulate controlled states. Strictly speaking, we verify the condition $\llbracket C \rrbracket |0\rangle = \sum_{x \in \{0,1\}^n} |x\rangle$ by

checking if $\boxed{C} = \underbrace{\text{green circle} \dots \text{green circle}}_n$. The diagram corresponding to the encoded state is simply $\boxed{D} = \boxed{C}$.

We will also extensively use the controlled scalars for the values $\frac{1}{\sqrt{2}}$ and 2.

Lemma 9.2.2 ([Jeandel et al., 2019]) Diagrams $C_{1/\sqrt{2}} = \text{diagram}$ and $C_2 = \text{diagram}$ are controlled states such that

$$C_{1/\sqrt{2}} = \text{diagram} = \text{diagram} \quad \text{and} \quad C_2 = \text{diagram} = \text{diagram} \quad (9.11)$$

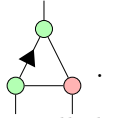
Controlled states have nice properties that allows to perform element-wise addition and tensor product of corresponding vectors:

Lemma 9.2.3 (Sum and tensor product [Jeandel et al., 2019]) For any controlled states $C_1, C_2 : 1 \rightarrow n$ and $C_3 : 1 \rightarrow m$ the diagrams:

$$C_+ = \text{diagram} \quad \text{and} \quad C_\otimes = \text{diagram} \quad (9.12)$$

are controlled states such that $\llbracket C_+ \rrbracket |1\rangle = \llbracket C_1 \rrbracket |1\rangle + \llbracket C_2 \rrbracket |1\rangle$ and $\llbracket C_\otimes \rrbracket |1\rangle = \llbracket C_1 \rrbracket |1\rangle \otimes \llbracket C_3 \rrbracket |1\rangle$.

We note that the lemma 7.4.14 states precisely the commutativity property of the gadget

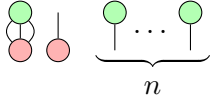


With the lemma 9.2.3 we can add two controlled states. More precisely, if one of the controlled states encodes a state diagram $D_1 : 0 \rightarrow n$ and the second controlled state encodes $D_2 : 0 \rightarrow n$

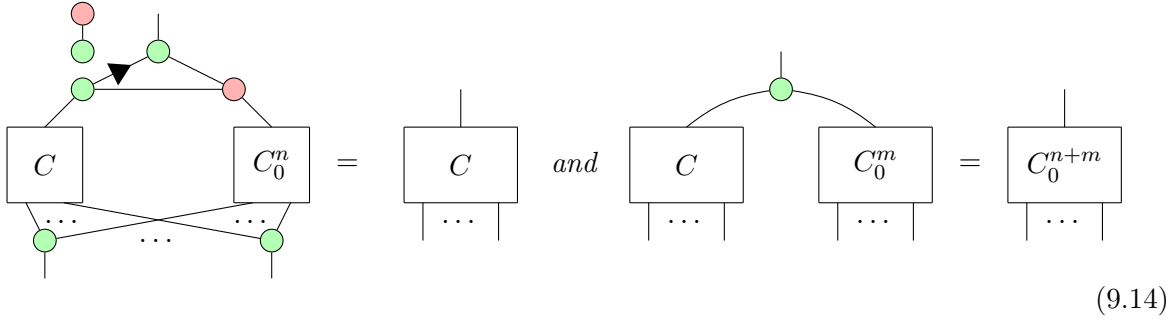
the diagram $\begin{smallmatrix} \text{diagram} \\ \boxed{D_+} \\ \text{diagram} \end{smallmatrix} = \begin{smallmatrix} \text{diagram} \\ \boxed{C_+} \\ \text{diagram} \end{smallmatrix}$ correspond to the matrix $\llbracket D_1 \rrbracket + \llbracket D_2 \rrbracket$. Moreover, as the construction returns a controlled state we can directly proceed to the addition of new terms. For N diagrams D_1, \dots, D_N encoded by the controlled versions C_1, \dots, C_N the diagram

$$D = \text{diagram} \quad (9.13)$$

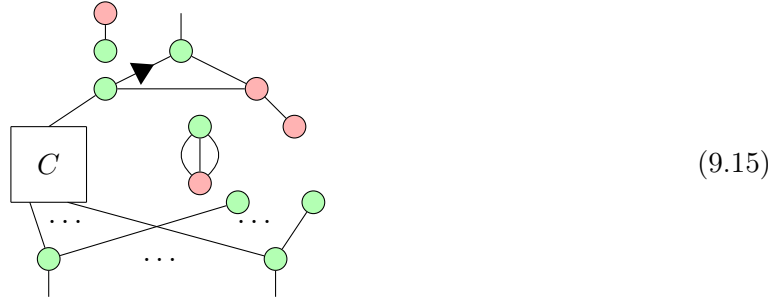
correspond to the matrix $\llbracket \mathbf{D} \rrbracket = \llbracket \mathbf{D}_1 \rrbracket + \cdots + \llbracket \mathbf{D}_N \rrbracket$.

We remark that for each number of output wires n the zero-element $C_0^n : 1 \rightarrow n$ is a controlled state . Indeed, we can prove the following lemma:

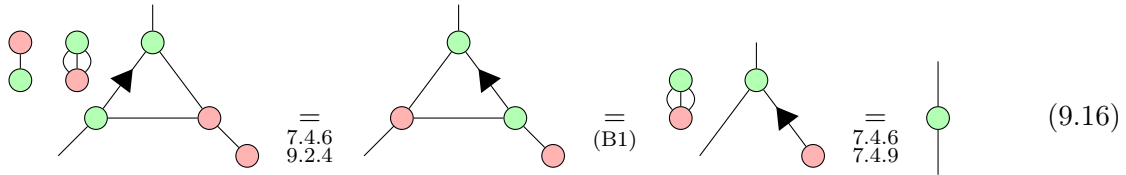
Lemma 9.2.4 *For any controlled state $C : 1 \rightarrow n$*



Proof: We start by proving the first equality. The left-hand side is:

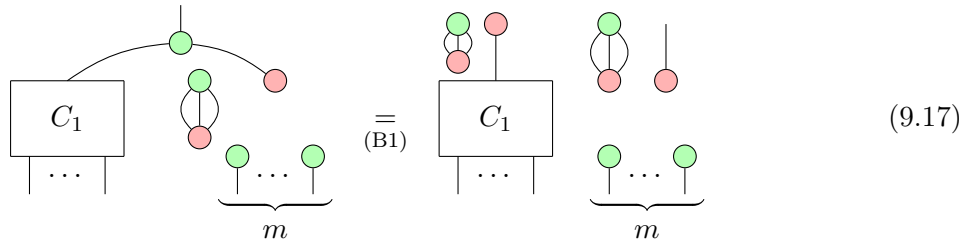


The top part of the diagram can be transformed as follows:



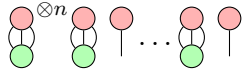
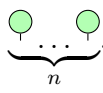
The statement follows from (S2) and (S1).

For the second equality we have:



The lemma follows from the definition of the controlled state. ■

The work [Yeung *et al.*,] suggests another definition of the controlled state. Essentially, the difference is the output of the zero-case. In their definition a controlled state $\tilde{C} : 1 \rightarrow n$,

when applied to the zero-input is transformed to  instead of the uniform superposition . Diagrammatically, the condition may be written as:

$$\begin{array}{c} \text{green} \\ \text{red} \\ \text{green} \\ \vdots \end{array} \boxed{\tilde{C}} = \begin{array}{c} \text{green} \\ \text{red} \end{array} \otimes n \begin{array}{c} \text{red} \\ \text{green} \end{array} \dots \begin{array}{c} \text{red} \\ \text{green} \end{array} \quad (9.18)$$

9.2.2 Controlizer

The lemma 9.2.3 provides a way to obtain a sum of two diagrams in a controlled state form. In order to extend the addition to *arbitrary* diagrams we introduce *controlizers* - maps that associate diagrams with the corresponding controlled states. In what follows we denote by $\text{ZX}[n, m]$ a family of ZX-diagrams with n inputs and m outputs.

The formal definition of the controlizer is:

Definition 9.2.5 (Controlizer) We say that a map $C : \text{ZX}[n, m] \rightarrow \text{ZX}[1, n + m]$ that associates to every diagram $D : n \rightarrow m$ a diagram $C(D) : 1 \rightarrow n + m$ is controlizer if the following conditions hold for any ZX-diagram D :

- $C(D)$ is a controlled state

•

$$\llbracket D \rrbracket = \left[\begin{array}{c} \otimes n + m \\ \text{green} \\ \text{red} \end{array} \begin{array}{c} n \\ \dots \end{array} \begin{array}{c} \text{red} \\ \text{green} \end{array} \pi \boxed{C(D)} \begin{array}{c} \dots \\ m \end{array} \right] \quad (9.19)$$

There are several maps that satisfy the definition of controlizer. For instance, the map presented in [Jeandel *et al.*, 2019] that associates to each diagram its *normal form* is in fact a controlizer. For the alternatively defined controlled state (9.18), the work [Yeung *et al.*,] shows that for each $D : 0 \rightarrow n$ the corresponding controlled state $\tilde{C}(D) : 1 \rightarrow n$ can be constructed from 2^n coefficient of the algebraic normal form.

In our work [Jeandel *et al.*, 2022] we suggested an approach that doesn't rely on normal forms. In our approach the controlizer is defined by induction over the composition and the tensor product. Importantly, the computational complexity of our inductive procedure is polynomial on the number of elementary terms in the decomposition of the diagram on composition and tensor products of elementary generators. As a result, the final diagram has polynomial size on the number of generators and wires in the input diagram.

Definition 9.2.6 (Inductive controlizer) We define the map $C : \text{ZX}[n, m] \rightarrow \text{ZX}[1, n + m]$ that associates to each diagram $D : n \rightarrow m$ a diagram $C(D) : 1 \rightarrow n + m$:

- For the generators β , π , \square , \cap , \cup , \times :

$$\begin{aligned}
 C(\beta) &= \text{diagram with } \beta \text{ and } \pi \text{ nodes}, & C(\pi) &= \text{diagram with } \pi \text{ node}, & C(\square) &= \text{diagram with } \square \text{ node}, \\
 C(\cup) &= C(\cap) = \text{diagram with } \cup \text{ node}, & C(\times) &= \text{diagram with } \times \text{ node}
 \end{aligned} \tag{9.20}$$

- Generators α and α' can be decomposed as follows using the above generators:

$$C\left(\begin{array}{c} n \\ \vdots \\ \alpha \\ \vdots \\ m \end{array}\right) = C\left(\begin{array}{c} \alpha \\ \vdots \\ n+m \end{array}\right), \quad C\left(\begin{array}{c} n \\ \vdots \\ \alpha' \\ \vdots \\ m \end{array}\right) = C\left(\begin{array}{c} \alpha' \\ \vdots \\ n+m \end{array}\right)$$

- For tensor product $D_{\otimes} = D_2 \otimes D_1$ and composition $D_{\circ} = D_3 \circ D_1$ where $D_1 : n \rightarrow m$, $D_2 : k \rightarrow l$ and $D_3 : m \rightarrow k$:

$$C(D_{\otimes}) = \text{diagram for } D_{\otimes}, \quad C(D_{\circ}) = \text{diagram for } D_{\circ} \tag{9.21}$$

We remark that the output of the inductive controlizer is not unique as it depends on the decomposition order.

Lemma 9.2.7 *The inductive controlizer from the definition (9.2.6) satisfies the definition (9.2.5) of controlizer.*

Proof: [Proof (lemma 9.2.7)] We prove the claim for the generators β , π , \square , \cap , \cup , \times and both compositions. The case of α and α' is a direct consequence of the axiom (S1).

First, we inductively prove that for all diagrams D the diagram $C(D)$ is controlled state.

- **Generators:**

$$C(\beta) \circ \text{diagram} = \text{diagram} \stackrel{(B1)}{=} \text{diagram} \stackrel{9.2.2, 7.4.12}{=} \text{diagram} \stackrel{7.4.6}{=} \text{diagram} \tag{9.22}$$

[illegible]

$$C\left(\begin{array}{c} \text{ } \\ \text{ } \end{array}\right) \circ \begin{array}{c} \text{ } \\ \text{ } \end{array} = \begin{array}{c} \text{ } \\ \text{ } \end{array} \stackrel{\text{(B1)}}{=} \begin{array}{c} \text{ } \\ \text{ } \end{array} \stackrel{7.4.6}{=} \begin{array}{c} \text{ } \\ \text{ } \end{array} \stackrel{9.2.2}{=} \begin{array}{c} \text{ } \\ \text{ } \end{array} \stackrel{7.4.19}{=} \begin{array}{c} \text{ } \\ \text{ } \end{array} \stackrel{\text{(H)}}{=} \begin{array}{c} \text{ } \\ \text{ } \end{array} \quad (9.24)$$

$$C(\cap) \circ \begin{array}{c} \text{red circle} \\ | \\ \text{green circle} \end{array} = \begin{array}{c} \text{red circle} \quad \text{red circle} \\ | \qquad | \\ \text{green circle} \quad \text{green circle} \end{array} \xrightarrow{\pi} \begin{array}{c} \text{red circle} \quad \text{red circle} \\ | \qquad | \\ \text{green circle} \quad \text{green circle} \end{array} \xrightarrow{(B1)} \begin{array}{c} \text{red circle} \quad \text{red circle} \\ | \qquad | \\ \text{green circle} \quad \text{green circle} \end{array} \xrightarrow[7.4.9]{9.2.2} \begin{array}{c} \text{red circle} \quad \text{green circle} \\ | \qquad | \\ \text{green circle} \end{array} \xrightarrow[7.4.6]{(B1)} \begin{array}{c} \text{green circle} \quad \text{green circle} \end{array} \quad (9.25)$$

[illegible]

- **Tensor product:** it directly follows from the lemma (9.2.3) that if $C(D_1)$ and $C(D_2)$ are controlled states, then the diagram $C(D_1 \otimes D_2)$ from (9.2.6) is controlled state.
- **Composition:** let's assume that $C(D_1)$ and $C(D_2)$ are controlled states. We show that $C(D_1 \circ D_2)$ is also a controlled state:

$$\begin{aligned}
 C(D_1 \circ D_2) \circ \text{red node} &= \text{Diagram with } C(D_1) \text{ and } C(D_3) \text{ blocks, } n \text{ and } k \text{ inputs, and } 2m \text{ outputs} \\
 &\stackrel{(B1)}{=} \text{Diagram with } C(D_1) \text{ and } C(D_3) \text{ blocks, } n \text{ and } k \text{ inputs, and } 2m \text{ outputs} \\
 &= \text{Diagram with } n \text{ and } k \text{ inputs, and } 2m \text{ outputs}
 \end{aligned}$$

$$\stackrel{7.4.6}{=} \underbrace{\text{---}\overset{\text{---}}{\circ}\text{---}\overset{\text{---}}{\circ}\text{---}\overset{\text{---}}{\circ}}_n \underbrace{\text{---}\overset{\text{---}}{\circ}\text{---}\overset{\text{---}}{\circ}\text{---}\overset{\text{---}}{\circ}}_k \quad (9.27)$$

Now we show that the map C satisfy the property (A.33), i.e. it correctly encodes the input diagram.

- Generators:


$$C(\beta) \circ (\text{diagram}) = \text{(B1)} \stackrel{\text{9.2.2}}{\underset{\text{7.4.10}}{=}} \text{diagram} \stackrel{\text{7.4.6}}{=} \text{diagram} \quad (9.28)$$

[illegible]

$$C \left(\begin{array}{c} | \\ \square \end{array} \right) \circ \begin{array}{c} \text{red circle} \\ \text{green circle} \end{array} \pi = \begin{array}{c} \text{diagram with red and green nodes and phase circles} \\ \text{9.2.2} \\ \text{7.4.18} \end{array} \stackrel{\text{(B1)}}{=} \begin{array}{c} \text{diagram with red and green nodes and phase circles} \\ \text{7.4.6} \end{array} \stackrel{\text{(S2)}}{=} \begin{array}{c} \text{diagram with a single green node and a phase circle} \\ \text{7.4.6} \end{array} \quad (9.30)$$

$$C(\cap) \circ \text{diagram} = \text{diagram} \stackrel{(B1)}{=} \text{diagram} \stackrel{\substack{9.2.2 \\ 7.4.11}}{=} \text{diagram} \stackrel{\substack{(S2) \\ 7.4.6}}{=} \text{diagram} \quad (9.31)$$

[illegible]

We obtain the property (A.33) for generators by swapping directions of outputs and multiplying by  \otimes^{n+m}

- **Composition:** Let's assume that the property (A.33) holds for diagrams $D_1 : n \rightarrow m$ and $D_2 : m \rightarrow k$.

$$\begin{aligned}
 & \text{Diagram 1: } \text{Inputs } n, k \text{ (green and red nodes) connect to a box } C(D_1 \circ D_2) \text{ with output } m+k. \\
 & = \text{Diagram 2: } \text{Inputs } n, k \text{ connect to boxes } C(D_1) \text{ and } C(D_2) \text{ with output } m+k. \\
 & = \text{Diagram 3: } \text{Inputs } n, k \text{ connect to boxes } D_1 \text{ and } D_2 \text{ with output } m+k. \\
 & = D_2 \circ D_1 \quad (9.33)
 \end{aligned}$$

- **Tensor product:** Let's assume that the property (A.33) holds for diagrams $D_1 : n \rightarrow m$ and $D_2 : k \rightarrow l$.

$$\begin{aligned}
 & \text{Diagram 1: } \text{Inputs } n+k, m+l \text{ connect to a box } C(D_2 \otimes D_1) \text{ with output } n+m+k+l. \\
 & = \text{Diagram 2: } \text{Inputs } n+k, m+l \text{ connect to boxes } C(D_2) \text{ and } C(D_1) \text{ with output } n+m+k+l. \\
 & = \text{Diagram 3: } \text{Inputs } n+k, m+l \text{ connect to boxes } D_2 \text{ and } D_1 \text{ with output } n+m+k+l. \\
 & = D_2 \otimes D_1 \quad (9.34)
 \end{aligned}$$

■

To give a flavor of our computation process we show how to obtain a controlled version of π .

$$C \left(\pi \right) = \begin{array}{c} \text{Diagram 1} \end{array} \stackrel{\substack{7.4.13 \\ 7.4.6 \\ (S2)}}{=} \begin{array}{c} \text{Diagram 2} \end{array} \stackrel{7.4.20}{=} \begin{array}{c} \text{Diagram 3} \end{array} \stackrel{\substack{7.4.16 \\ 7.4.6}}{=} \begin{array}{c} \text{Diagram 4} \end{array} \quad (9.35)$$

Addition of arbitrary diagrams

The controlizer allows to map every two ZX-diagrams $D_1 : n \rightarrow m$ and $D_2 : n \rightarrow m$ to controlled states $C_1 : 1 \rightarrow n + m$ and $C_2 : 1 \rightarrow n + m$. The lemma 9.2.3 provides a way to obtain a sum of resulting controlled states. By combining this two results we can recover the diagram for addition of D_1 and D_2 . The addition protocol is resumed in the following theorem:

Theorem 9.2.8 For diagrams $D_1 : n \rightarrow m$ and $D_2 : n \rightarrow m$ the diagram

$$D_+ = \begin{array}{c} \text{Diagram 1} \end{array} \stackrel{\substack{n \\ \dots}}{\text{Diagram 2}} \text{ where } C_+ = \begin{array}{c} \text{Diagram 3} \end{array}$$

is such that $\llbracket D_+ \rrbracket = \llbracket D_1 \rrbracket + \llbracket D_2 \rrbracket$.

Proof: [Proof (theorem 9.2.8)] The theorem follows from the definition of controlizer and 9.2.3.

■

We illustrate the diagrammatic addition with a simple example.

Example 9.2.9 Using 9.2.8, we construct a diagram D corresponding to the addition of \cap and π .

$$D = \begin{array}{c} \text{Diagram 1} \end{array} \stackrel{\substack{7.4.6 \\ (B1) \\ 7.4.12}}{=} \begin{array}{c} \text{Diagram 2} \end{array} \stackrel{\substack{7.4.8 \\ 7.4.13}}{=} \begin{array}{c} \text{Diagram 3} \end{array} \quad (9.36)$$

$$\stackrel{\substack{(B2) \\ 7.4.16}}{=} \begin{array}{c} \text{Diagram 4} \end{array} \stackrel{7.4.18}{=} \begin{array}{c} \text{Diagram 5} \end{array} \stackrel{\substack{(B1) \\ 7.4.6}}{=} \begin{array}{c} \text{Diagram 6} \end{array} \quad (9.37)$$

In this example we simplified the diagram with graphical rewriting to obtain a known result:

$$\llbracket \cap \rrbracket + \llbracket \pi \rrbracket = (|00\rangle + |11\rangle) + (|01\rangle + |10\rangle) = 2|++\rangle = \llbracket \begin{array}{c} \text{Diagram 7} \end{array} \rrbracket \quad (9.38)$$

9.2.3 Relation to other results

As a matter of fact, our inductive procedure leads to large diagrams, even when the initial diagrams are fairly simple. Moreover, resulting diagrams don't resemble the initial ones. This defect is however consistent with the intuition. Indeed, contrary to sequential and parallel compositions, the addition is not a physical operation, hence it is not surprising that can't be easily incorporated in the framework of ZX-diagrams. On the positive side, we can rely on the powerful equation theory of the ZX-calculus to simplify, when it is possible, the diagrams representing the sum of diagrams.

An alternative approach for the addition in the algebraic ZXW-calculus was presented in [Yeung *et al.*,]. It also relies on the controlled forms of the diagram. We remark that the circuit protocol (9.2) for addition of unitaries presented in [Childs and Wiebe, 2012] also requires controlled versions of unitary transformation.

The controlled forms used in [Yeung *et al.*,] are different from ours. Notably, contrary to our approach that accepts diagrams with an arbitrary number of input and output wires, the controlled forms in [Yeung *et al.*,] are defined only for two families of diagrams.

The first family consists of state diagrams $\text{ZX}[0, n]$. The work [Yeung *et al.*,] suggests to construct a controlled state from the *normal form* [Wang, 2020]. Importantly, for a state diagram with n outputs the normal form has an exponential size on n .

The second family of diagrams treated in [Yeung *et al.*,] includes diagrams $\text{ZX}[m, m]$ that correspond to *square matrices*. A procedure suggested in [Yeung *et al.*,] for controlled matrices requires the decomposition of the initial diagram D on a product of diagrams for so-called *elementary matrices* [Wang and Yeung, 2021]. As far as we have understood, in the general case the decomposition on elementary matrices requires an explicit computation of the matrix corresponding to the diagram. Therefore, although for given controlled forms the addition from [Yeung *et al.*,] is done diagrammatically with the help of *W-spider*, the computation of the controlled forms pass by the semantics. In contrast, in our inductive procedure the entire process is maintained in the graphical framework.

However, in some special cases of practical importance the approach suggested in [Yeung *et al.*,] is beneficial. For instance, as was shown in [Yeung *et al.*,] for local Hamiltonians made out of Pauli tensors the decomposition on elementary matrices is straightforward. As a consequence, the controlled versions are easy to compute and the overall procedure leads to nice-looking diagrams. Moreover, considering the diagrams from more expressive algebraic ZXW-calculus leads to a simpler representation of linear combinations of diagrams with arbitrary complex coefficients.

9.3 Differentiation of ZX-diagrams

A basic definition of the *derivative* is the derivative for a smooth function $f : \mathbb{R} \rightarrow \mathbb{R}$:

$$\frac{\partial f}{\partial x}(x_0) = \lim_{\delta \rightarrow 0} \frac{f(x_0 + \delta) - f(x_0)}{\delta} \quad (9.39)$$

For a function on multiple variables this definition is extended to a *gradient*. In the traditional calculus the derivative usually is computed from the decomposition of the function $f : \mathbb{R} \rightarrow \mathbb{R}$ on a sum, product and composition of elementary functions. For elementary functions the derivatives are explicitly computed from the definition (9.39). Derivatives for complex functions are computed from the linearity and the product rules. The *linearity* and the *product rules* that are derived from the definition.

A parameterized ZX-diagram $D(\beta)$ can be understood as a function $f : \mathbb{R} \rightarrow \text{ZX}$ that associated to each real value $\beta \in \mathbb{R}$ a ZX-diagram. In the context of variational algorithms, we consider the derivatives over the *parameters*. We remark that as the sums and subtractions are unnatural for ZX-diagrams we avoid to define the derivative as a limit of a finite difference (9.39).

In order to keep the notations simple in what follows we restrict our attention to one-variable diagrams. Moreover, we will consider only the family of *linear diagrams* (7.5). In our definition of linear diagrams the phases are affine functions with integer coefficients before parameters. We denote the corresponding matrices by $\mathcal{M}(\beta)$. The elements of matrices in $\mathcal{M}(\beta)$ belong to the ring generated by complex numbers and the smooth functions $e^{i\mathcal{A}}$ where $\mathcal{A} = \{k\beta \mid k \in \mathbb{Z}\}$ is the group of affine functions with integer coefficients.

The sum for the matrices in $\mathcal{M}(\beta)$ is naturally defined by entrywise addition. Therefore, we can define the derivative in $\mathcal{M}(\beta)$ by specifying the general definition for monoidal categories with sums given in [Toumi *et al.*, 2021]:

Definition 9.3.1 (Derivative in $\mathcal{M}(\beta)$) *A derivative ∂_M in $\mathcal{M}(\beta)$ is a linear unary operator on $\mathcal{M}(\beta)$ that associated to a matrix $L \in \mathcal{M}(\beta)$, $L : n \rightarrow m$ another matrix $L' \in \mathcal{M}(\beta)$, $L' : n \rightarrow m$ of the same dimensions. It satisfies the following axioms:*

$$\circ\text{-product rule} : \partial_M[A \circ B] = \partial_M[A] \circ B + A \circ \partial_M[B]$$

$$\otimes\text{-product rule} : \partial_M[A \otimes B] = \partial_M[A] \otimes B + A \otimes \partial_M[B]$$

Rather than defining the derivative as a limit of finite difference and deriving the product rules, the work [Toumi *et al.*, 2021] suggests an axiomatic definition that proclaims the *product rules* as the fundamental property of the derivative operator. Furthermore, the work [Toumi *et al.*, 2021] shows how their general definition can be used to define derivatives for *bags of ZX-diagrams*.

Although our theorem (9.2.8) provides a fully diagrammatic way for addition of diagrams, we avoid to extend the language with a formal introduction of a sum operator in order to circumvent unnecessary complications. Therefore, the general definition from [Toumi *et al.*, 2021] doesn't apply. Instead, we suggest an alternative semantics for the derivative in the ZX-calculus. In place of product rules we require derivative to satisfy the property of *diagrammatic differentiation*. This property states the coherence between the derivative of a diagram and the derivative of the corresponding linear map:

Definition 9.3.2 (ZX-derivative) *A derivative $\partial_{\text{ZX}} : \text{ZX}[n, m] \rightarrow \text{ZX}[n, m]$ is an unary operator that commutes with the standard interpretation:*

$$\llbracket \partial_{\text{ZX}} D \rrbracket = \partial_M \llbracket D \rrbracket \quad (9.40)$$

where ∂_M is the derivative (9.3.1) in $\mathcal{M}(\beta)$.

We remark that even if we restricted the consideration to diagrams over one variable, all results may be easily extended to the case of partial derivatives ∂_{β_i} for linear diagrams with an arbitrary number of variables.

9.3.1 Diagrammatic differentiation with controlizers

The derivative in $\mathcal{M}(\beta)$ is defined through product rules that involve sums. In this section we show how to incorporate these sums in the diagrammatic framework using *controlizers*.

In what follows we denote by C any map that satisfies the definition (9.2.5) of controlizer.

all possible outputs are semantically equivalent and by completeness of ZX-calculus are equivalent as diagrams.

By combining C -derivatives and controlizers we can reproduce the semantics of product rules in a graphical representation.

Definition 9.3.4 Given a C -derivative Δ , let $\partial_C : \text{ZX}(\beta) \rightarrow \text{ZX}(\beta)$ be the unary operator such that for any diagram $D : n \rightarrow m$

$$\partial_C[D] = \begin{array}{c} \text{Diagram with } n+m \text{ inputs and } m \text{ outputs} \\ \text{where the first } n \text{ inputs are grouped by a brace labeled } n \\ \text{and the last } m \text{ inputs are grouped by a brace labeled } m \\ \text{The diagram is labeled } \Delta(D) \end{array} \quad (9.45)$$

Theorem 9.3.5 The operator ∂_C satisfies the definition (9.40) of diagrammatic differentiation.

Proof: [Proof (theorem 9.3.5)] We give the proof by induction over tensor product and composition.

- **Generators:** Firstly, we show that $\partial_M \left[\left[\begin{array}{c} \beta \\ \vdots \end{array} \right] \right] = ie^{i\beta} |-\rangle$ and $\partial_M \left[\left[\begin{array}{c} -\beta \\ \vdots \end{array} \right] \right] = -ie^{-i\beta} |-\rangle$.

Indeed, we have:

$$\partial_M \left[\left[\begin{array}{c} \beta \\ \vdots \end{array} \right] \right] = \partial_M (|+\rangle + e^{i\beta} |-\rangle) = ie^{i\beta} |-\rangle \quad (9.46)$$

$$\partial_M \left[\left[\begin{array}{c} -\beta \\ \vdots \end{array} \right] \right] = \partial_M (|+\rangle + e^{-i\beta} |-\rangle) = -ie^{-i\beta} |-\rangle \quad (9.47)$$

For the diagrams $\partial_C \left(\begin{array}{c} \beta \\ \vdots \end{array} \right)$ and $\partial_C \left(\begin{array}{c} -\beta \\ \vdots \end{array} \right)$ we verify the property of diagrammatic differentiation:

$$\left[\left[\begin{array}{c} \pi \\ \vdots \end{array} \right] \right] = \left[\left[\begin{array}{c} \pi \\ \vdots \end{array} \right] \right] = ie^{i\beta} |-\rangle \quad (9.48)$$

$$\left[\left[\begin{array}{c} \pi \\ \vdots \end{array} \right] \right] = \left[\left[\begin{array}{c} \pi \\ \vdots \end{array} \right] \right] = -ie^{-i\beta} |-\rangle \quad (9.49)$$

For a generator $g : n \rightarrow m$ that doesn't depend on β :

$$\left[\partial_C(g) \right] = \left[\left[\begin{array}{c} \pi \\ \vdots \end{array} \right] \right] \stackrel{(S1)}{=} \left[\left[\begin{array}{c} \pi \\ \vdots \end{array} \right] \right] = (0)_{n \times m} \quad (9.50)$$

- **Tensor product:** Let's assume that $[\partial_C(D_1)] = \partial_M[D_1]$ and $[\partial_C(D_2)] = \partial_M[D_2]$ for $D_1 : n \rightarrow m$ and $D_2 : l \rightarrow k$. For $D = D_1 \otimes D_2$:

$$[\partial_C(D)] = \left[\begin{array}{c} \text{Diagram with } \Delta(D) \text{ box and inputs } n+k, m+l \end{array} \right] \quad (9.51)$$

By applying the definition (9.3.3) of the C -derivative for the tensor product we obtain the diagram

$$\left[\begin{array}{c} \text{Diagram with } \Delta(D_1), C(D_2), C(D_1), \Delta(D_2) \text{ boxes and inputs } n, k, m, l \end{array} \right] \quad (9.52)$$

We denote by $\begin{array}{c} T \\ \hline \dots \quad \dots \\ \hline n+m \quad k+l \end{array}$ the part of the diagram surrounded by the red frame. According to the definition, C -derivatives $\Delta(D_1)$, $\Delta(D_2)$ and controller outputs $C(D_1)$, $C(D_2)$ are controlled states. Therefore, from the lemma 9.2.3 we get:

$$\left[\begin{array}{c} T \\ \hline \dots \quad \dots \\ \hline n+m \quad k+l \end{array} \right] = \left[\begin{array}{c} C(D_1) \quad \Delta(D_2) \\ \hline \dots \quad \dots \end{array} \right] + \left[\begin{array}{c} \Delta(D_1) \quad C(D_2) \\ \hline \dots \quad \dots \end{array} \right] \quad (9.53)$$

Meanwhile, by bending wires we see that the resting part of the diagram (9.52) is:

$$B = \left[\begin{array}{c} \text{Diagram with inputs } n, k, m, l \end{array} \right] \quad (9.54)$$

The sum in $\mathcal{M}(\beta)$ is distributive over the composition. Therefore,

$$\left[\begin{array}{c} \text{Diagram with } T \text{ box and inputs } n, k, m, l \end{array} \right] = \quad (9.55)$$

$$\left[\begin{array}{c} \text{Diagram 1} \\ \text{Diagram 2} \end{array} \right] \quad (9.56)$$

$$+ \left[\begin{array}{c} \text{Diagram 3} \\ \text{Diagram 4} \end{array} \right] \quad (9.57)$$

The claim follows from the induction assumption and the definition of controlizer.

- **Composition:** Following the same reasoning as for the tensor product we obtain for $D_1 : n \rightarrow m$ and $D_2 : m \rightarrow l$

$$\partial_C[D_2 \circ D_1] = \text{Diagram } T \quad (9.58)$$

where the diagram T is exactly the same as in (9.53). On more time, by distributivity of the sum in $\mathcal{M}(\beta)$ over the composition we obtain

$$\llbracket \partial_C[D_2 \circ D_1] \rrbracket = \text{Diagram 5} \quad (9.59)$$

$$+ \text{Diagram 6} \quad (9.60)$$

The claim follows from the definition of controlizer and the induction assumption. ■

We consider a simple example to illustrate how to perform the inductive procedure.

Example 9.3.6 We apply the definition (9.3.4) to the simple scalar diagram $\text{Diagram 7} = \text{Diagram 8}$.

Notice that $C\left(\text{Diagram 9}\right) = \text{Diagram 10} = \text{Diagram 11}$, moreover $C(\text{Diagram 12})$ was

already found in 9.35. We obtain a diagram for $C \left(\begin{array}{c} \square \quad \pi \quad \square \\ | \quad | \quad | \end{array} \right)$:

$$(9.61)$$

We know from 9.2.4 that $\Delta \left(\begin{array}{c} \square \quad \pi \quad \square \\ | \quad | \quad | \end{array} \right) = \begin{array}{c} \text{green node} \quad \text{green node} \\ | \quad | \end{array}$. By definition, $\Delta \left(\begin{array}{c} -\beta \quad \beta \\ | \quad | \end{array} \right) =$

and $\partial_C \left(\begin{array}{c} -\beta \quad \beta \\ | \quad | \end{array} \right) =$

9.3.2 Formula for derivatives in $\text{ZX}(\beta)$

Although perfectly correct, the differentiation procedure described above leads to very puzzling output even for small diagrams (see the example 9.3.6). In this section we provide a simpler approach to obtain the derivative of a diagram in $\text{ZX}(\beta)$ written in a special form. We formalize it in definitions ∂_{ZX} and ∂_P of unary operators that satisfy the property of diagrammatic differentiation (9.40).

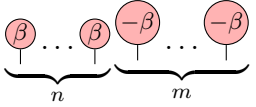
Firstly, we remark that even in the inductive procedure the effort can be significantly reduced if we separate the parts that depend on the parameter from the rest of the diagram that is constant on β . Indeed, for the constant diagrams $A : k \rightarrow l$ and $B : m \rightarrow k$ the C -derivative is the controlled version of the zero diagram C_0^{k+l} and C_0^{m+k} . Therefore, for a parameterized diagram $D(\beta) : n \rightarrow m$ the lemma 9.2.4 implies :

$$(9.62)$$

$$(9.63)$$

This expressions are significantly simple that the general case (9.3.3) where both parts of the compositions depend on β .

We want to further explore this property to find simplified formulas for the derivative. Let's

denote by $X_\beta(n, m)$ diagrams  from $ZX(\beta)$.

From the (S1) and (H) rules and the paradigm *Only topology matters* follows:

Claim 9.3.7 *Using the rules of ZX calculus, each diagram $D(\beta)$ from $ZX(\beta)$ may be transformed into the form*

$$D(\beta) = \begin{array}{c} \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array} \quad \begin{array}{|c|} \hline X_\beta(n, m) \\ \hline \dots \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline D_2 \\ \hline \dots \\ \hline \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array} \quad \begin{array}{|c|} \hline \beta \dots \beta -\beta \dots -\beta \\ \hline \dots \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline D_2 \\ \hline \dots \\ \hline \end{array} \end{array} \quad (9.64)$$

where n, m are some integer numbers and D_1, D_2 are constant with respect to β . We call diagrams in this form β -factored.

A rigorous demonstration of the claim 9.3.7 may be found in [Jeandel *et al.*, 2018b].

The diagram $X_\beta(n, m)$ has a particularly regular form, so we expect it to have a nice-looking derivative. We would like to develop the intuition from the semantics $\partial_M \llbracket X_\beta(n, m) \rrbracket$.

Intuition

We denote by $|\beta\rangle = |0\rangle + e^{i\beta}|1\rangle$ the state $\llbracket \begin{array}{|c|} \hline \beta \\ \hline \end{array} \rrbracket$ and by $|\!-\!\beta\rangle = |0\rangle + e^{-i\beta}|1\rangle$ the state $\llbracket \begin{array}{|c|} \hline -\beta \\ \hline \end{array} \rrbracket$.

We also introduce the notation:


$$|\beta(n, m)\rangle = |\beta\rangle^{\otimes n} \otimes |\!-\!\beta\rangle^{\otimes m} = \sum_{x \in \{0,1\}^{n+m}} e^{i\beta(|x^+| - |x^-|)} |x^+\rangle \otimes |x^-\rangle \quad (9.65)$$

where $x^+ \in \{0,1\}^n$ corresponds to first n qubits that are in the state $|\beta\rangle^{\otimes n}$ and $x^- \in \{0,1\}^m$ corresponds to the last m qubits that are in the state $|\!-\!\beta\rangle^{\otimes m}$. The notation $|x| : \{0,1\}^k \rightarrow \mathbb{Z}_+$ stands for the Hamming weight of the binary string x .

By applying entrywise differentiation to the vector $|\beta(n, m)\rangle$, we obtain the following result:

$$\partial_M (|\beta(n, m)\rangle) = \sum_{x \in \{0,1\}^{n+m}} i(|x^+| - |x^-|) e^{i\beta(|x^+| - |x^-|)} |x^+\rangle \otimes |x^-\rangle \quad (9.66)$$

We observe that $\partial_M (|\beta(n, m)\rangle) = iM_\Delta |\beta(n, m)\rangle$ where $M_\Delta : 2^{n+m} \rightarrow 2^{n+m}$ is a linear map such that $M_\Delta : |x\rangle \rightarrow (|x^+| - |x^-|)|x\rangle$.

In order to find a diagram D_M for the matrix M_Δ we introduce the controlled-triangle $D_\lambda : 2 \rightarrow 2$. Semantically, we want the diagram D_λ to apply the triangle  to the second input if the first one is in the $|1\rangle$ state and do nothing otherwise. We denote the matrix corresponding

As the original input wires are used only in control, the output of such procedure is $\alpha_x |b_0, \dots, b_{n+m}\rangle$.

This procedure can be directly translated in a diagram D_λ ;

$$D_\lambda = \begin{array}{c} \overbrace{\quad n \quad} \quad \overbrace{\quad m \quad} \\ \text{Diagram with } n \text{ and } m \text{ control wires and } \Lambda \text{ gates} \end{array} = \begin{array}{c} \text{Diagram with } \pi \text{ gates and control wires} \end{array} \stackrel{(B1)}{=} \begin{array}{c} \text{Diagram with } \pi \text{ gates and control wires} \end{array} \quad (9.72)$$

7.4.11
7.4.10
7.4.16

The diagram for the derivative of $X_\beta(n, m) = \beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta}$ can be obtained by conjugating every input with Hadamard boxed:

$$\begin{aligned} \partial_M \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] &= \partial_M \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] = \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] \circ \partial_M \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] \quad (9.73) \\ &= \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] \circ \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] = \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] \circ \left[\left[\beta \dots \beta \overset{\circ}{\beta} \dots \overset{\circ}{\beta} \right] \right] \quad (9.74) \end{aligned}$$

By applying the rule (H) to the final we obtain

$$\partial_M [X_\beta(n, m)] = \left[\left[\text{Diagram with } \pi \text{ gates and control wires} \right] \right] \quad (9.75)$$

Diagrammatic differentiation of β -factored forms

Motivated by our the simplicity of the result (9.75), we suggest an alternative non-inductive definition for derivatives of diagrams in β -factored forms:

Definition 9.3.8 Given a diagram $D(\beta) = D_2 \circ (D_1 \otimes X_\beta(n, m))$ in β -factored form, let $\partial_{ZX}[D] = D_2 \circ (D_1 \otimes \partial_{ZX}[X_\beta(n, m)])$ where

$$\partial_{\text{ZX}}[X_\beta(n, m)] = \partial_{\text{ZX}} \left[\underbrace{\beta \dots \beta}_n \underbrace{-\beta \dots -\beta}_m \right] \quad (9.76)$$

$$= \text{diagram with } n+m \text{ green circles and } m \text{ red circles} \quad (9.77)$$

Although our definition is inspired from the intuitive reflection presented above, we provide a rigorous diagrammatic demonstration of the soundness of our definition. More precisely, we prove the following theorem:

Theorem 9.3.9 *The operator $\partial_{\text{ZX}}[-]$ defined at (9.3.8) satisfies the property of diagrammatic differentiation: for any diagram $D(\beta) \in \text{ZX}(\beta)$ in β -factored form,*

$$\llbracket \partial_{\text{ZX}} D(\beta) \rrbracket = \partial_M \llbracket D(\beta) \rrbracket \quad (9.78)$$

We remark that according to the definition 9.3.2 the derivative $D' : n \rightarrow m$ of a diagram $D : n \rightarrow m$ that is constant on β is such that $\llbracket D' \rrbracket = \partial_M \llbracket D \rrbracket = (0)_{n \times m}$. Therefore, the theorem 9.3.9 is a direct consequence of the following lemma:

Lemma 9.3.10 *For any n, m :*

$$\llbracket \partial_{\text{ZX}} X_\beta(n, m) \rrbracket = \partial_M \llbracket X_\beta(n, m) \rrbracket \quad (9.79)$$

Proof: [Proof (lemma 9.3.10)] We prove the lemma by induction. The demonstration is done for $n = n + 1$, the proof for $m = m + 1$ is directly obtainable in the same way.

Base: We already know that $\partial_M \llbracket \beta \rrbracket = ie^{i\beta} |-\rangle$. Thus, for the case $n = 1, m = 0$ the lemma statement follows from:

$$\llbracket \partial_{\text{ZX}} X_\beta(1, 0) \rrbracket = \left[\text{diagram} \right] \stackrel{7.4.25}{=} \left[\text{diagram} \right] \stackrel{7.4.6}{=} ie^{i\beta} |-\rangle \quad (9.80)$$

Step: By induction, we assume that the equation (9.79) holds for some n and m . We show that under this assumption $\llbracket \partial_{\text{ZX}} X_\beta(n + 1, m) \rrbracket = \partial_M \llbracket X_\beta(n + 1, m) \rrbracket$.

In order to prove the induction we will proceed in three steps. Firstly we apply the product rule to the matrix $\partial_M \llbracket X_\beta(n + 1, m) \rrbracket$:

$$\begin{aligned} \partial_M \llbracket X_\beta(n + 1, m) \rrbracket &= \partial_M \llbracket X_\beta(1, 0) \rrbracket \otimes \llbracket X_\beta(n, m) \rrbracket + \llbracket X_\beta(1, 0) \rrbracket \otimes \partial_M \llbracket X_\beta(n, m) \rrbracket \\ &= \llbracket \partial_{\text{ZX}} [X_\beta(1, 0)] \rrbracket \otimes \llbracket X_\beta(n, m) \rrbracket + \llbracket X_\beta(1, 0) \rrbracket \otimes \partial_{\text{ZX}} \llbracket X_\beta(n, m) \rrbracket \end{aligned} \quad (9.81)$$

where the second equality follows from the inductive assumption.

Secondly, we show the last sum in the equation (9.81) can be represented diagrammatically with a controlled state, i.e. the following claim holds:

Claim 9.3.11 *We can find a controlled state $\tilde{X} : 1 \rightarrow n + 1 + m$ and a constant scalar $c \in \text{ZX}_{\frac{\pi}{2}}$ such that*

$$\left[c \begin{array}{c} \text{diagram} \\ \tilde{X} \\ \dots \end{array} \right] = \llbracket \partial_{\text{ZX}} [X_\beta(1, 0)] \rrbracket \otimes \llbracket X_\beta(n, m) \rrbracket + \llbracket X_\beta(1, 0) \rrbracket \otimes \partial_{\text{ZX}} \llbracket X_\beta(n, m) \rrbracket \quad (9.82)$$

Finally, we will show that we can transform the diagram $\partial_{ZX} X_\beta(n+1, m)$ to $c \left[\begin{array}{c} \text{green circle} \quad \pi \\ \text{red circle} \quad \pi \\ \hline \tilde{X} \\ \vdots \end{array} \right]$

with a sequence of graphical rewrites:

Claim 9.3.12

$$\partial_{ZX} X_\beta(n+1, m) = c \left[\begin{array}{c} \text{green circle} \quad \pi \\ \text{red circle} \quad \pi \\ \hline \tilde{X} \\ \vdots \end{array} \right] \quad (9.83)$$

As the rewrite rules preserve the semantics, it follows from the claims above that:

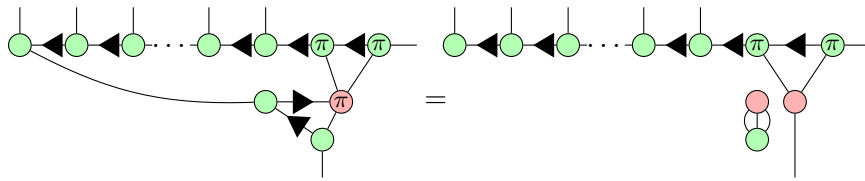
$$\begin{aligned} \llbracket \partial_{ZX} X_\beta(n+1, m) \rrbracket &\stackrel{9.3.12}{=} c \left[\begin{array}{c} \text{green circle} \quad \pi \\ \text{red circle} \quad \pi \\ \hline \tilde{X} \\ \vdots \end{array} \right] \\ &\stackrel{9.3.11}{=} \llbracket \partial_{ZX} [X_\beta(1, 0)] \otimes X_\beta(n, m) \rrbracket + \llbracket X_\beta(1, 0) \otimes \partial_{ZX} [X_\beta(n, m)] \rrbracket \\ &\stackrel{9.81}{=} \partial_M \llbracket X_\beta(n+1, m) \rrbracket \end{aligned} \quad (9.84)$$

■

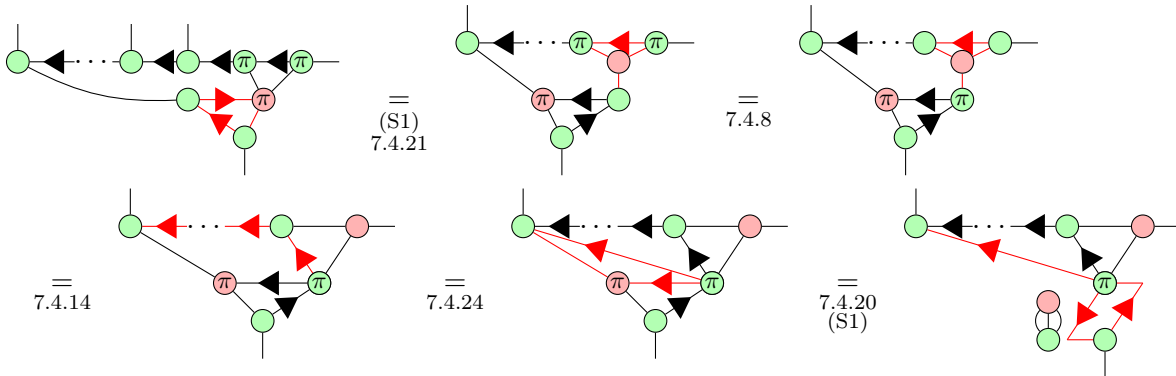
Proofs

While proving the claims 9.3.11 and 9.3.12 we will repeatedly use the following lemmas:

Lemma 9.3.13

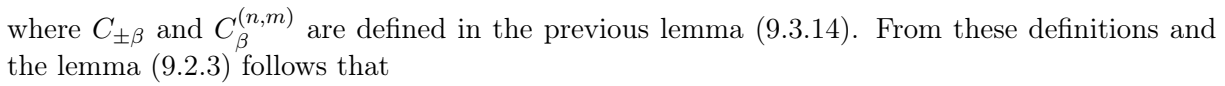


Proof: [lemma 9.3.13] The right hand side transformation:




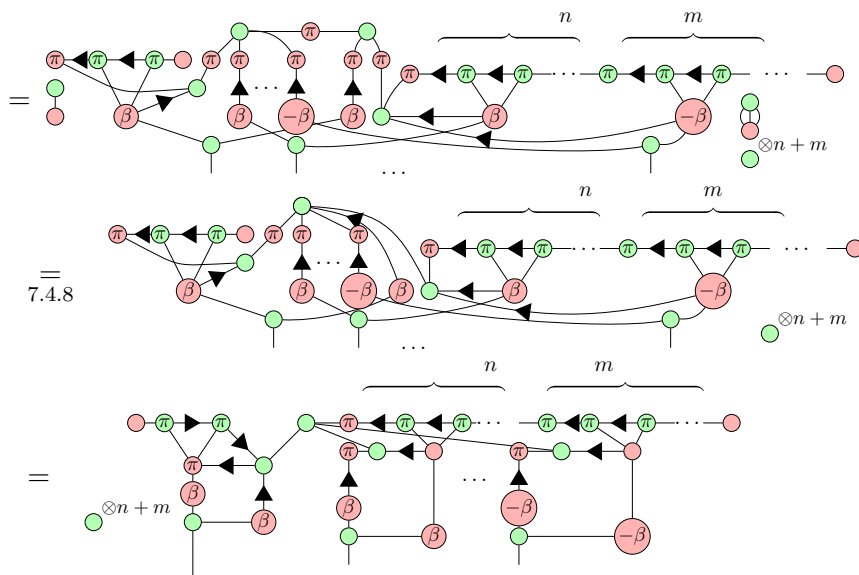


We introduce the diagram:



Therefore the claim holds for the controlled state (9.88) and the scalar $c = \frac{\pi}{2} \otimes (n+m-2)$ ■

Proof: [claim 9.3.12] Firstly, we simplify the expression $\tilde{X} \circ$ :



$$\begin{aligned}
& \stackrel{7.4.13}{=} \stackrel{7.4.8}{=} \text{Diagram 1} \\
& \stackrel{7.4.26}{=} \text{Diagram 2} = \textcircled{1}
\end{aligned}
\tag{9.90}$$

Diagram 1: A ZX-diagram with a green spider labeled \otimes^{n+m} on the left. It has two main sections: a green section with n spiders and a red section with m spiders. The green section has spiders with parameters π and β . The red section has spiders with parameters π and $-\beta$. Diagram 2: The same diagram as Diagram 1, but with a red frame around the green section and a red frame around the red section. The red section is now labeled m and the green section is labeled n . The red section has spiders with parameters π and $-\beta$. The green section has spiders with parameters π and β . The entire diagram is equal to $\textcircled{1}$.

We apply the lemma 9.3.13 to parts in the red frame and obtain:

$$\begin{aligned}
& \textcircled{1} = \text{Diagram 3} \\
& \stackrel{7.4.13}{=} \text{Diagram 4}
\end{aligned}
\tag{9.91}$$

Diagram 3: A ZX-diagram with a green spider labeled \otimes^{n+m} on the left. It has two main sections: a green section with n spiders and a red section with m spiders. The green section has spiders with parameters π and β . The red section has spiders with parameters π and $-\beta$. Diagram 4: The same diagram as Diagram 3, but with a red frame around the green section and a red frame around the red section. The red section is now labeled m and the green section is labeled n . The red section has spiders with parameters π and $-\beta$. The green section has spiders with parameters π and β . The entire diagram is equal to $\textcircled{1}$.

The claim holds for the scalar $c = \frac{\pi}{2} \otimes^{n+m-2}$ ■

9.3.3 Simplified formula for paired spiders

Variational quantum algorithms use gradients in the search for optimal parameter values. The objective minimized by these algorithms can be expressed as $\langle \psi(\beta) | H | \psi(\beta) \rangle$ where the diagram for $\langle \psi(\beta) | = (|\psi(\beta)\rangle)^\dagger$ is obtained out of the diagram for $|\psi(\beta)\rangle$ by flipping up side down followed by the change of signs in spiders. Therefore, parameters in the diagram for $\langle \psi(\beta) | H | \psi(\beta) \rangle$ appear in pairs $\begin{array}{c} -\beta \\ \beta \end{array}$.

We suggest a more compact formula for diagrams in what we call *pair-factored form*: $D_2 \circ (D_1 \otimes Y(n))$. In this expression $Y_\beta(n) = \underbrace{\begin{pmatrix} -\beta & \beta \end{pmatrix} \dots \begin{pmatrix} -\beta & \beta \end{pmatrix}}_n$.

Lemma 9.3.15 *The diagram:*

$$\partial_P(Y_\beta(n)) = \text{Diagram 5}
\tag{9.92}$$

Diagram 5: A ZX-diagram with a green spider labeled \otimes^3 on the left. It has a green section with n spiders and a red section with n spiders. The green section has spiders with parameters π and β . The red section has spiders with parameters π and $-\beta$. The entire diagram is equal to $\partial_P(Y_\beta(n))$.

satisfies $\llbracket \partial_P(Y_\beta(n)) \rrbracket = \partial_M \llbracket Y_\beta(n) \rrbracket$.

We prove the lemma (9.3.15) by applying the same approach as in the proof of the formula (9.3.10) for individual spiders. We can then replace by (9.92) the expression (9.77) in 9.3.8 and obtain the derivative for diagrams in pair-factored form.

Proof: [lemma 9.3.15]

Base: We have $\partial_M \left[\left[\begin{array}{c} \text{red circle } -\beta \\ \text{red circle } \beta \end{array} \right] \right] = \partial_M(|++\rangle + e^{i\beta}|+-\rangle + e^{-i\beta}|-+\rangle + |--\rangle) = i(e^{i\beta}|+-\rangle - e^{-i\beta}|-+\rangle).$

Simplifying the diagram (9.92) for $n = 1$ we obtain:

$$\partial_P(Y(1)) = \begin{array}{c} \text{diagram with green circles } \pi, \pi/2 \text{ and red circles } -\beta, \beta \end{array} \stackrel{\substack{(B1) \\ 7.4.13 \\ 7.4.9}}{=} \begin{array}{c} \text{diagram with green circles } \pi, \pi/2 \text{ and red circles } \pi-\beta, \beta \end{array} \quad (9.93)$$

From the definition of the standard interpretation $\left[\left[\begin{array}{c} \text{green circle } \pi \end{array} \right] \right] = |00\rangle - |11\rangle = |+-\rangle + |-+\rangle$ and $[R_X(\pi - \beta) \otimes R_X(\beta)] \circ [|+-\rangle + |-+\rangle] = e^{i\beta}|+-\rangle + e^{-i\beta}|-+\rangle$. Therefore, the lemma is true for $n = 1$.

Step: By induction, we assume that $\llbracket \partial_P Y_\beta(n) \rrbracket = \partial_M \llbracket Y_\beta(n) \rrbracket$. We show that under this assumption the lemma holds for $n = n + 1$ by proving following claims:

Claim 9.3.16

$$\partial_M \llbracket Y_\beta(n+1) \rrbracket = \llbracket \partial_P Y_\beta(1) \otimes Y(n) \rrbracket + \llbracket Y_\beta(1) \times \partial_P Y(n) \rrbracket \quad (9.94)$$

Claim 9.3.17 *There exist a controlled state \tilde{Y} and a scalar c such that*

$$\left[c \otimes \left(\tilde{Y} \circ \begin{array}{c} \text{red circle } \pi \\ \text{green circle } \uparrow \end{array} \right) \right] = \llbracket \partial_P Y_\beta(1) \otimes Y_\beta(n) \rrbracket + \llbracket Y_\beta(1) \times \partial_P Y_\beta(n) \rrbracket \quad (9.95)$$

Claim 9.3.18 $ZX \vdash \partial_P(Y_\beta(n+1)) = c \otimes \left(\tilde{Y} \circ \begin{array}{c} \text{red circle } \pi \\ \text{green circle } \uparrow \end{array} \right)$ where C and c are respectively the controlled state and scalar from the previous claim.

The claim (9.3.16) follows from the product rule for ∂_M and from the induction assumption.

Proof: [Proof (claim 9.3.17)] We can explicitly check that for each $n \in \mathbb{N}$ the following

diagram is a controlled state: $\tilde{Y}(n) = \begin{array}{c} \text{diagram with green circles } \pi, \pi/2 \text{ and red circles } -\beta, \beta \end{array}$ and $\tilde{Y}(n)|1\rangle =$

$$\left[\left[\begin{array}{c} \text{green circle } \pi \\ \text{red circle } -\pi/2 \end{array} \right] \otimes^{2n-2} \partial_P[Y_\beta(n)] \right]:$$

Indeed,

$$\begin{array}{c} \text{diagram with green circles } \pi, \pi/2 \text{ and red circles } -\beta, \beta \end{array} = \begin{array}{c} \text{diagram with green circles } \pi, \pi/2 \text{ and red circles } -\beta, \beta \end{array}$$

$$\begin{aligned}
&= \text{Diagram 1} \otimes \text{Diagram 2} = \text{Diagram 3} \\
&\tilde{Y}(n) = \text{Diagram 4} \\
&\stackrel{(B1) \ 7.4.10}{=} \text{Diagram 5}
\end{aligned}$$

We denote by \tilde{Y} the diagram $\tilde{Y} = \tilde{Y}(1) \begin{matrix} C_{-\beta} & C_{+\beta} & \dots & C_{-\beta} & C_{+\beta} & C_{-\beta} & C_{+\beta} \end{matrix} \tilde{Y}(n)$ where C_{β} and $C_{-\beta}$ were defined in the lemma (9.3.14). Due to the lemma (9.2.3),

$$\begin{aligned}
\llbracket \hat{C} \rrbracket |1\rangle &= \left[\begin{array}{c} \pi \\ \text{Diagram} \end{array} \partial_P [Y(1)] \right] \otimes \underbrace{\left[\begin{array}{c} \otimes^2 \\ \text{Diagram} \end{array} \right] \dots \left[\begin{array}{c} \otimes^2 \\ \text{Diagram} \end{array} \right]}_n \\
&+ \left[\begin{array}{c} \otimes^2 \\ \text{Diagram} \end{array} \right] \otimes \left[\begin{array}{c} \pi \\ \text{Diagram} \end{array} \otimes^{2n-2} \partial_P [Y(n)] \right] \\
&= \left[\begin{array}{c} \pi \\ \text{Diagram} \end{array} \otimes^{2n} \right] \left[\llbracket \partial_P Y(1) \otimes Y(n) \rrbracket + \llbracket Y(1) \times \partial_P Y(n) \rrbracket \right] \quad (9.96)
\end{aligned}$$

Therefore, the claim holds for $C = \hat{C}$ and $c = \begin{array}{c} \pi \\ \text{Diagram} \end{array} \otimes^{2n}$ ■

Proof: [Proof (claim 9.3.18)] Firstly, we simplify the diagrams $\tilde{Y} \circ \begin{array}{c} \pi \\ \text{Diagram} \end{array}$:

$$\begin{aligned}
\tilde{Y} \circ \begin{array}{c} \pi \\ \text{Diagram} \end{array} &= \text{Diagram 1} \\
&\stackrel{7.4.8}{=} \text{Diagram 2}
\end{aligned}$$

We subsequently apply the lemma (9.3.13) to parts in the red frames:

The last expression multiplied by the constant $c = \begin{array}{c} \pi \\ \pi/2 \end{array} \begin{array}{c} \beta \\ \beta \end{array}^{2n}$ is equal to $\partial_P Y(n+1)$. Induction is proven. ■

It is possible to extend 9.3.15 to find the derivative for $X_\beta(n, m)$ when $n \neq m$. Indeed, using the fact that $\begin{array}{c} \pm\beta \\ \beta \end{array} = \begin{array}{c} \pi \\ \pi/2 \end{array}$ we can balance the number of β and $-\beta$. For instance, if $n > m$:

Example 9.3.19 We apply 9.3.15 to the same diagram as in 9.3.6:

We observe that using the formula for the pair-factored form (9.92) we obtain a much more compact result than with the inductive procedure (see the example 9.3.6). Even if both approaches lead to diagrams with the same semantics, in practice the diagrams obtained with the formulas (9.92) and (9.77) are less verbose. For this reason they are easier to manipulate.

9.3.4 Relation to other results

Contrary to derivatives defined in [Toumi *et al.*, 2021] and [Zhao and Gao, 2021] in our approaches a derivative of a ZX-diagram is another ZX-diagram. In other words, we circumvent the necessity of introducing formal sums. In the inductive approach we proceed by integration of the product rules using *controlizers*.

We also observe that as the derivative of a constant diagram is trivial, it is beneficial to ‘factor-out’ the part that depends on the parameter. We adopted this approach in formulas for diagrams in β -factored form and pair-factored forms. To derive the formulas, we got the intuition in the desired semantics of the diagrams. Thenceforth, our formulas were rigorously proven by induction.

A result similar to our simplified formula for β -factored forms was independently derived in [Wang and Yeung, 2022]. The major difference between our formula and the method shown in [Wang and Yeung, 2022] is the *considered language*. Indeed, in our work we operate ZX-diagrams while Wang and Yeung consider diagrams from more expressive *algebraic ZX-calculus*.

The difficulty to represent derivatives for non-linear diagrams follows from the fact that there is no a simple way to represent real numbers in the vanilla ZX-calculus. In the algebraic ZX-calculus this restriction is removed. As a consequence, when an algebraic ZX-diagram is parameterized by an arbitrary derivable function $f(x)$, the differentiated algebraic ZX-diagram is parametrised by $f'(x)$.

9.4 Diagrammatic representation of Ising Hamiltonians

Traditionally, in the variational algorithms the optimization objective is encoded by a *Hamiltonian*. For combinatorial optimization problems the Hamiltonian is typically an Ising model:

$$H = \sum_{i=1}^n h_i Z_i + \sum_{1 \leq i < j \leq n} h_{i,j} Z_i Z_j \quad (9.101)$$

There is no direct way to transform the definition of the Hamiltonian (9.101) to a ZX-diagram. Indeed, Hamiltonian is a non-unitary matrix equal to a *sum* of Pauli gates that is inherently difficult to represent as a diagram.

The work [Yeung *et al.*,] suggests a convenient way to represent Hamiltonians that are linear combinations of Pauli tensors in *algebraic ZXW-calculus*. In [Jeandel *et al.*, 2022] we adopted a different approach that use our diagrammatic differentiation together with the *Stone’s theorem*. The Stone theorem relating the derivative of a unitary group $e^{i\gamma H}$ to its generator H was used before in [Toumi *et al.*, 2021]. However, in [Toumi *et al.*, 2021] the derivatives were defined as formal sums of diagrams. As a consequence, the work [Toumi *et al.*, 2021] didn’t suggest a single diagram representation for a Hamiltonian but rather a representation as a sum of diagrams.

We remark that for an Ising Hamiltonians H the diagram $D_U(\beta)$ of the linear map $U(\beta) = e^{i\beta H}$ is easy to find. For Hamiltonians with integer coefficients the matrix $U(\beta) = e^{i\beta H}$ belongs to $\mathcal{M}(\beta)$. It satisfies the definition of strongly continuous one-parameter unitary group:

$$h = \begin{array}{c} \textcircled{\pi} \\ \textcircled{-\frac{\pi}{2}} \end{array} [\partial_{\text{ZX}} D_{U(\beta)}]_{\beta \rightarrow 0} = \begin{array}{c} \textcircled{\pi} \\ \textcircled{\pi} \end{array} \quad (9.105)$$

Conclusion

In this thesis we explored the potential of quantum algorithms for combinatorial optimization on realistic industrial applications. On the theoretical side, we have developed a set of tools that allows to reason about variational algorithms in ZX-calculus.

We have considered two usecases issued from the field of smart charging. For both usecases our presentation contains *i)* a detailed complexity analysis of the natural formulations *ii)* a modelization in a quantum-amenable formulation *iii)* a description of approximation properties of the resulting models. We provide experimental protocols for QAOA and RQAOA for this applications. In particular, we specify selected QUBO encodings and parameter optimization routines. Finally, we numerically evaluate the performance of our routines. We compare our evaluations to the performance of classical algorithms. We carefully justify the choice of classical counterparts based on approximability properties of the considered problems.

We believe that our presentation positively contributes in **the establishment of a general methodology** for analysis of quantum algorithms for optimization problems. Completed with a mapping to hardware specifications, it constitutes a fairly exhaustive protocol for the examination of quantum heuristics applied to a considered practical problem.

Quantum algorithms for combinatorial optimization that are analyzed in this work are *heuristics*. Therefore, their practical performance in general heavily depends on the chosen application, or, more precisely, on considered instances of the optimization problem. For this reason it is extremely important to **select right usecases**. After working on this thesis, we may claim that a well-suited usecase for quantum algorithms should be at the same moment *simple enough* to be easily modeled as *unconstrained problem over binary variables* but *hard enough* to be difficult for classical algorithms. The model simplicity is crucial to avoid the overconsumption of limited resources available on NISQ devices. The classical intractability is necessary to leave the room for quantum advantage.

For an optimization problem the *classical intractability* may be interpreted in different ways. In this work we highlighted the importance of considering **approximation algorithms and heuristics rather than exponential exact procedures** for a fair evaluation of the difficulty of NP-hard optimization problems. In addition, we have observed that the instances issued from practical applications may differ in approximation complexity with the general model selected to represent them.

From the algorithmic side we have detected the importance of **the fine-tuning of the experimental protocol**. For instance, we have confirmed the observation stating that the genuine bottleneck of QAOA is the search of optimal parameters. This problem can be addressed by exploring the properties of the parameter landscape. In our numerical experiments, we have tested several existing approaches for the parameter optimization and confirmed the intuition behind them.

Our numerical results are reported in the sections 4.4 and 5.3.3 proved the interest of the

application of quantum algorithms for both considered usecases. **We have observed that QAOA and RQAOA have the performance that is comparable with pertinent classical competitors.** We remark that due to the lack of resources our numerical analysis was restricted to versions of QAOA with low depth. We believe that the increase of depth may improve the performance of QAOA making it a powerful heuristic for practical applications.

In order to extend the range of applications that can be addressed by quantum algorithms we considered hybrid quantum-classical approaches. We believe that **hybridization is essential** to leverage the power of quantum computers for industrial applications. In particular, we suggested to outsource a difficult part of computation in the classical Branch & Price to a quantum routine. The resulting method, called **Quantum-assisted Branch & Price**, can be used to address integer linear programs with many variables. We have observed that quantum routines can enhance the runtime performance of the original classical scheme but the actual improvement is *not* the same across different families of considered instances (see section 6.4.3). We expect that on a real hardware the performance dependence on the instance properties (in particular on the connectivity) will be even more significant [Harrigan *et al.*, 2021, Pagano *et al.*, 2020].

A further research in this direction requires more computational resources, in particular a real hardware of plausible size.

Future work

Our work points out several promising directions for the future research.

In the close-term perspective we can **enhance our experimental protocols** by improving their performance and reducing the runtime. In particular, we can leverage the parameter concentration phenomena [Brandao *et al.*, 2018] and integrate learning-based techniques in the parameter optimization routine. We can start by considering the approaches presented in [Khairy *et al.*, 2020, Moussa *et al.*, 2022].

In the long-term we are planning to validate our results on the **quantum hardware** and integrate hardware specifications in the design of experimental protocols.

Indeed, a fair evaluation of the performance of quantum algorithms requires hardware experimentations. In this direction we are planning to evaluate the **impact of noise** and the limited connectivity on the performance of quantum algorithms. In addition, we intend to explore the *error mitigation* techniques to handle the imperfections of the NISQ hardware.

On the application side, it is particularly compelling to **qualify our hybrid Quantum-assisted Branch & Price algorithm** on other practical optimization problems. For instance, in our future work we intend to apply this approach to the *Frequency Containment Reserve* problem - an industrial problem issued from the field of energy management where pricing sub-problems take the form of *Shortest Path* to be found in graphs representing the charge/discharge of electrical vehicles connected to the grid.

Finally, we aim to continue the *theoretical exploration of variational algorithms* by using diagrammatic reasoning. For this purpose we lean on our tools for **addition and differentiation of ZX-diagrams**. For instance, we can use diagrammatic representation to analyze such properties of the parameter landscape as symmetries and barren plateau. We believe that with some extra work we can improve the protocol design for variational algorithms, for example by suggesting new parameter-shift rules. In addition, we plan to explore how such fundamental properties as *QAOA supremacy* can be understood in the diagrammatic terms.

Appendix A

Linear programming

Linear program is an optimization problem over *continuous* variables such that the objective function and the constraints are *linear* on input variables. While not strictly belonging to the field of combinatorial optimization, linear programs are ubiquitous in subroutines of exact and approximate algorithms.

Formally, a linear program in a standard form is:

$$\min \sum_{i=1}^n c_i x_i \tag{A.1}$$

$$\text{s.t. } \sum_{i=1}^n a_{j,i} x_i \geq b_j, \quad j \in [1, \dots, m] \tag{A.2}$$

$$x_i \geq 0, \quad i \in [1, \dots, m] \tag{A.3}$$

We remark that any linear program can be reformulated in a standard form.

Although relatively simple to formulate, linear program is an extremely powerful model for the *operational research*. For instance, it incorporates many resource allocation problems. A seminal example of a linear program is the *diet problem* where one has to build a meal of minimal cost satisfying nutritional requirements.

From the mathematical point of view a bounded linear program is precisely the optimization over a *convex polyhedron* defined by the set of constraints. Crucially, if the instance is feasible and bounded there exists at least one extreme point of the feasible region that corresponds to the optimal solution. We remark that in general case the optimal solution isn't unique.

A.1 Duality

For each linear program there exists another linear program called *dual* that results from an attempt to provide a lower bound on the minimum value of the initial problem. In the context of duality the initial linear program is usually referred as *primal problem* or *primal formulation*.

To formulate the dual problem we introduce one dual variable λ_j per inequality constraint (A.2) from the primal formulation. We want to derive bounds on the variables λ_j such that for each feasible solution of the primal problem we have:

$$\sum_{j=1}^m b_j \lambda_j \leq \sum_{i=1}^n c_i x_i \tag{A.4}$$

Firstly, we impose the non-negativity on the variables λ_j . The non-negativity of λ_j allows to state that:

$$\lambda_j \sum_{i=1}^n a_{j,i} x_i \geq b_j \lambda_j \quad (\text{A.5})$$

Taking the sum over all constraints we obtain the condition:

$$\sum_{j=1}^m \lambda_j \sum_{i=1}^n a_{j,i} x_i \geq \sum_{j=1}^m b_j \lambda_j \quad (\text{A.6})$$

or, equivalently,

$$\sum_{i=1}^n \left(\sum_{j=1}^m a_{j,i} \lambda_j \right) x_i \geq \sum_{j=1}^m b_j \lambda_j \quad (\text{A.7})$$

We remark that for non-negative primal variables $x_i \geq 0$ the condition (A.4) is satisfied if for each variable $x_i \geq 0$ the respective coefficients on the left sides of (A.7) are smaller or equal to the coefficients on the right side, i.e.

$$\sum_{j=1}^m a_{j,i} \lambda_j \leq c_i \quad \implies \quad \sum_{j=1}^m b_j \lambda_j \leq \sum_{i=1}^n \left(\sum_{j=1}^m a_{j,i} \lambda_j \right) x_i \leq \sum_{i=1}^n c_i x_i \quad (\text{A.8})$$

Therefore, if dual variables λ_j satisfy the constraints

$$\sum_{j=1}^m a_{j,i} \lambda_j \leq c_i, \quad \forall i \in [1, \dots, n] \quad (\text{A.9})$$

$$\lambda_j \geq 0, \quad \forall j \in [1, \dots, m] \quad (\text{A.10})$$

then $\sum_{j=1}^m b_j \lambda_j$ provides a lower bound for the objective value of the primal problem: $\sum_{i=1}^n c_i x_i$.

Roughly speaking, the *dual problem* consists in finding the best lower bound derivable in such way. Formally, we find the optimal bound by solving the following linear program:

$$\max \sum_{j=1}^m b_j \lambda_j \quad (\text{A.11})$$

$$\text{s.t. } \sum_{j=1}^m a_{j,i} \lambda_j \leq c_i, \quad \forall i \in [1, \dots, n] \quad (\text{A.12})$$

$$\lambda_j \geq 0, \quad \forall j \in [1, \dots, m] \quad (\text{A.13})$$

There is one dual variable for each constraint in the primal problem and every variable in the primal problem generates a constraint in the dual formulation. By construction, the optimal value of the dual formulation provides a lower bound on the optimum for the initial problem.

In fact, this latter observation about the respective optimal values of dual and primal formulations can be further strengthened with the *duality theorem* [Gass and Harris, 2001]. The duality theorem states in particular that if both problems are feasible and bounded then they have the *same* optimal value, i.e.

$$\sum_{j=1}^m b_j \lambda_j^* = \sum_{i=1}^n c_i x_i^* \quad (\text{A.14})$$

A.1.1 Complementary slackness

Another important relation of dual and primal formulations is resumed in the *theorem of complementary slackness*. Roughly speaking, the theorem states that in an optimal solution $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ either $x_i^* = 0$ or the corresponding constraint in the dual formulation is binding (and vice-versa).

This theorem is a consequence of the duality theorem. Firstly, we recall that by definition the optimal values of both formulations satisfy the inequality:

$$\sum_{j=1}^m b_j \lambda_j^* \leq \sum_{i=1}^n \left(\sum_{j=1}^m a_{i,j} \lambda_j^* \right) x_i^* \leq \sum_{i=1}^n c_i x_i^* \quad (\text{A.15})$$

Actually, for feasible and bounded problems it follows from the duality theorem that:

$$\sum_{i=1}^n \left(\sum_{j=1}^m a_{i,j} \lambda_j^* \right) x_i^* = \sum_{i=1}^n c_i x_i^* \quad (\text{A.16})$$

or, equivalently,

$$\sum_{i=1}^n \left(\sum_{j=1}^m a_{i,j} \lambda_j^* - c_i \right) x_i^* = 0 \quad (\text{A.17})$$

According to the definition a feasible solution for the primal satisfies $x_i^* \geq 0$ while for the feasible solution of the dual we have $\sum_{j=1}^m a_{i,j} \lambda_j^* \leq c_i$. Therefore, an optimal pair $\mathbf{x}^*, \boldsymbol{\lambda}^*$ is feasible if

$$\left(\sum_{j=1}^m a_{i,j} \lambda_j^* - c_i \right) x_i^* \leq 0 \quad (\text{A.18})$$

The theorem can be proved by contradiction. Indeed, if the complementary slackness doesn't hold, then from (A.17) and (A.18) follows that there exists at least one $v \in [1, \dots, n]$ such that $\left(\sum_{j=1}^m a_{v,j} \lambda_j^* - c_v \right) x_v < 0$. As the sum (A.17) is strictly equal to zero, there should be at least one $u \in [1, \dots, n]$ such that $\left(\sum_{j=1}^m a_{u,j} \lambda_j^* - c_u \right) x_u > 0$. Such value x_u violates the feasibility condition (A.18).

An equivalent result for the dual solution can be proven following the same reasoning.

A.2 Simplex method

The *simplex method* was originally suggested in 1951 by Dantzig. A detailed description can be found in [Nash, 2000]. An introduction and a comprehensive review of important theoretical and practical aspect of the simplex method can be found in [Vanderbei, 2014].

In a nutshell, the simplex algorithm iteratively considers extreme points of the polyhedron of feasible solutions. At each step it moves to an improving point. Usually, when the assignment $x_1 = \dots = x_n \equiv 0$ is feasible it is taken as a starting point. In the rest of the section we will consider the simplest case where the instance is bounded and a feasible initial solution is known.

More precisely, we begin by introducing slack variables s_1, \dots, s_m that transform inequality constraints (A.2) to equalities. For the simplex method it actually doesn't matter is the variable

is present in the initial formulation or if it was introduced to capture the slack differences. Therefore, we consider the initial variables and the slack variables together and we denote them as x_1, \dots, x_{n+m} .

With the slack variables the problem is reformulated as follows:

$$\min \chi(x_1, \dots, x_{n+m}) = \sum_{i=1}^n c_i x_i \quad (\text{A.19})$$

$$\text{s.t. } \sum_{i=1}^n a_{j,i} x_i - x_{n+j} = b_j, \quad j \in [1, \dots, m] \quad (\text{A.20})$$

$$x_i \geq 0, \quad i \in [1, \dots, n+m] \quad (\text{A.21})$$

We remark that by fixing values of any n variables we can derive a unique feasible solution from the equality constraints (A.20). This is the core idea of the simplex algorithm. In a nutshell, at each iteration the simplex methods chose a subset \mathcal{B} of n variables that will be set to zero. The set \mathcal{B} is called basis. We denote by \mathcal{N} the set of m non-basis variables. For fixed values of basis variables the values of variables in \mathcal{N} are computed from the equality constraints:

$$x_i = \alpha_i + \sum_{j \in \mathcal{B}} \omega_{j,e} x_j \quad (\text{A.22})$$

Likewise, by replacing the variables $x_i, i \in \mathcal{N}$ with the expressions (A.22) the objective function can be written as a linear combination of basis variables:

$$\chi_{\mathcal{B}}(x) = \chi_0 + \sum_{i \in \mathcal{B}} \chi_i x_i \quad (\text{A.23})$$

The simplex method searches the set \mathcal{B} for an *improving variable*. A variable $x_i \in \mathcal{B}$ is improving if we can diminish the value of the objective function by increasing the value of x_i from zero. Clearly, a variable is improving if $\chi_i < 0$.

In bounded problems the linear relationships (A.20) determine an upper bound on the potential increase of every improving variable that doesn't violate the constraints $x_i \geq 0, \forall i \in \mathcal{N}$. The strongest bound correspond to so-called *entering variable* $x_e \in \mathcal{N}$. In other words, for an improving variable x_i the entering variable is:

$$x_e = \alpha_e - \omega_{i,e} x_i + \sum_{\substack{j \in \mathcal{B} \\ j \neq i}} \omega_{j,e} x_j \quad (\text{A.24})$$

such that the ratio $u_{i,e} = \alpha_e / \omega_{i,e}$ is the smallest one for $x_e \in \mathcal{N}$. If we increase x_i for more than $u_{i,e}$ units the constraint $x_e \geq 0$ becomes violated.

We remark that there may be several improving variables. Usually, we select the one that leads to the most important decrease in the value of the objective function. Precisely, we chose the improving variable i with the value $\chi_i u_{i,e} < 0$ of largest magnitude.

Once the method has identified one improving and one entering variable it performs a *pivot*. During the pivot procedure the improving variable x_i leaves the basis replaced by the entering variable x_e . In other words, we rewrite the objective function (A.23) and the constraints (A.22) by replacing x_i with

$$x_i = \frac{1}{\omega_{i,e}} \left(\alpha_e - x_e + \sum_{\substack{j \in \mathcal{B} \\ j \neq i}} \omega_{j,e} x_j \right) \quad (\text{A.25})$$

The new solution is computed by fixing $x_e = 0$. The variable x_i gets the value $u_{i,e}$ and the value of the objective function is decreased by $|\chi_i u_{i,e}|$. The procedure continues until there is no more improving variables.

A.2.1 Graphical interpretation

The simplex procedure has a particularly nice graphical interpretation.

Firstly, we remark that setting $x_j = 0$ for a slack variable ($j \in [n+1, \dots, m]$) implies that the constraint j in the original formulation is *binding*, i.e. the solution lays exactly on the polyhedron edge defined by $\sum_{i=1}^n a_{j,i} = b_j$. Similarly, if $x_i = 0$ for a variable from the initial formulation the solution belongs to the edge defined by the constraint $x_i \geq 0$. Therefore, the basis \mathcal{B} defines an intersection of n polyhedron edges. For a polyhedron in n -dimensional space such intersection is precisely one of the extreme points. The graphical representation of the simplex method for the two-variable case is shown on the Figure A.1.

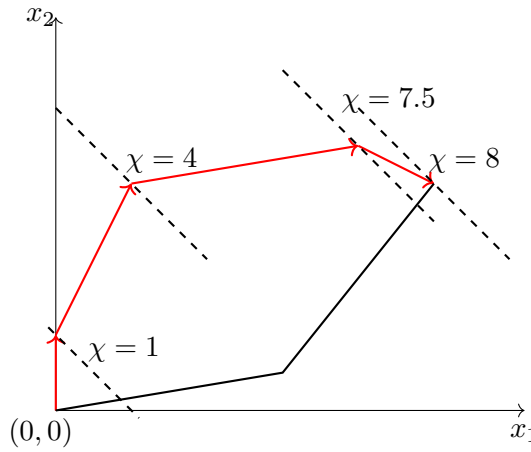


Figure A.1: At each iteration the simplex method moves to an extreme point corresponding to a better solution. Levels of the objective function $\chi(x_1, x_2) = x_1 + x_2$ are denoted by dashed lines.

We remark that if we add the integrality requirement $\mathbf{x} \in \mathbb{Z}^n$ on the variables (as in *Integer Linear Programs (ILP)* studied in the Section 6.1) the optimum solutions are in general *not* in the extreme point of the polyhedron. Indeed, if an extreme point has fractional coordinates it is not even a feasible solution for such formulations. This is the major reason why the optimization of linear programs with integer variables is in general much harder than the optimization over real values.

For bounded problems the polyhedron has $\binom{n+m}{n}$ extreme points. Therefore, in the worst case the simplex method has an exponential runtime. However, such situation rarely happens in practice and the simplex remains one of the most efficient methods for the linear programs. Alternatively, linear program can be solved in guaranteed polynomial runtime with so-called interior-point methods[Karmarkar, 1984, Khachiyan, 1980].

Bibliography

- [Aaronson and Gottesman, 2004] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70:052328, Nov 2004.
- [Aharonov *et al.*, 2007] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal on Computing*, 37(1):166–194, 2007.
- [Alaoui *et al.*, 2021] Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Local algorithms for maximum cut and minimum bisection on locally treelike regular graphs of large degree, 2021.
- [Albash and Lidar, 2018] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), jan 2018.
- [Albash *et al.*, 2017] Tameem Albash, Victor Martin-Mayor, and Itay Hen. Temperature scaling law for quantum annealing optimizers. *Physical Review Letters*, 119(11), sep 2017.
- [Alidaee *et al.*, 1994] Bahram Alidaee, Gary A. Kochenberger, and Ahmad Ahmadian. 0-1 quadratic programming approach for optimum solutions of two scheduling problems. *International Journal of Systems Science*, 25(2):401–408, 1994.
- [Alidaee *et al.*, 2008] Bahram Alidaee, Gary A. Kochenberger, Karen R. Lewis, Mark W. Lewis, and Haibo Wang. A new approach for modeling and solving set packing problems. *Eur. J. Oper. Res.*, 186:504–512, 2008.
- [Alkhamis *et al.*, 1998] Talal M. Alkhamis, Merza Hasan, and Mohamed A. Ahmed. Simulated annealing for the unconstrained quadratic pseudo-boolean function. *Eur. J. Oper. Res.*, 108:641–652, 1998.
- [Altshuler *et al.*, 2010] Boris Altshuler, Hari Krovi, and Jérémie Roland. Anderson localization makes adiabatic quantum optimization fail. *Proceedings of the National Academy of Sciences*, 107(28):12446–12450, 2010.
- [Ambainis, 2005] Andris Ambainis. Quantum search algorithms. *arXiv e-prints*, pages quant-ph/0504012, April 2005.
- [Amin, 2009a] M. H. S. Amin. Consistency of the adiabatic theorem. *Phys. Rev. Lett.*, 102:220401, Jun 2009.
- [Amin, 2009b] M. H. S. Amin. Consistency of the adiabatic theorem. *Phys. Rev. Lett.*, 102:220401, Jun 2009.

- [Andrade *et al.*, 2008] Diogo Andrade, Mauricio Resende, and Renato Werneck. Fast local search for the maximum independent set problem. volume 18, pages 220–234, 05 2008.
- [Anthony *et al.*, 2017] Martin Anthony, Endre Boros, Yves Crama, and Aritanan Gruber. Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming*, 162, 2017.
- [Arora and Kale, 2016] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *J. ACM*, 63(2), may 2016.
- [Arora and Safra, 1998] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *J. ACM*, 45(1):70–122, jan 1998.
- [Arora *et al.*, 1998] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, may 1998.
- [Arute *et al.*, 2019] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando Brandao, David Buell, Brian Burkett, Yu Chen, Jimmy Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Michael Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew Harrigan, Michael Hartmann, Alan Ho, Markus Rudolf Hoffmann, Trent Huang, Travis Humble, Sergei Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, Dave Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod Ryan McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin Jeffery Sung, Matt Trevithick, Amit Vainsencher, Benjamin Villalonga, Ted White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.
- [Ausiello *et al.*, 1999] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 1999.
- [ave, 2022] Baromètre national des infrastructures de recharge ouvertes au public, Sep 2022.
- [Babbush *et al.*, 2021] Ryan Babbush, Jarrod Ryan McClean, Michael Newman, Craig Michael Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, 2:010103, 2021.
- [Backens and Kissinger, 2019] Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. *Electronic Proceedings in Theoretical Computer Science*, 287:23–42, jan 2019.
- [Backens *et al.*, 2021] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale. *Quantum*, 5:421, mar 2021.

- [Backens, 2014] Miriam Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, sep 2014.
- [Balas and Xue, 1991] Egon Balas and Jue Xue. Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs. *SIAM Journal on Computing*, 20(2):209–221, 1991.
- [Balas and Yu, 1986] Egon Balas and Chang Sung Yu. Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.*, 15:1054–1068, 1986.
- [Balas *et al.*, 1993] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.*, 58:295–324, 05 1993.
- [Barahona, 1986] Francisco Barahona. A solvable case of quadratic 0–1 programming. *Discrete Applied Mathematics*, 13(1):23–26, 1986.
- [Barak and Marwaha, 2022] Boaz Barak and Kunal Marwaha. Classical algorithms and quantum limitations for maximum cut on high-girth graphs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [Barak *et al.*, 2015] Boaz Barak, Ankur Moitra, Ryan O’Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree, 2015.
- [Barends *et al.*, 2016] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, A. G. Fowler, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, P. J. J. O’Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, E. Solano, H. Neven, and John M. Martinis. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534(7606):222–226, jun 2016.
- [Barkoutsos *et al.*, 2020] Panagiotis Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using cvar. *Quantum*, 4:256, 04 2020.
- [Basso *et al.*, 2021] João Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou. The quantum approximate optimization algorithm at high depth for maxcut on large-girth regular graphs and the sherrington-kirkpatrick model. *ArXiv*, abs/2110.14206, 2021.
- [Basso *et al.*, 2022] Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou. The quantum approximate optimization algorithm at high depth for maxcut on large-girth regular graphs and the sherrington-kirkpatrick model. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [Battaglia *et al.*, 2005] Demian Battaglia, Giuseppe Santoro, and Erio Tosatti. Optimization by quantum annealing: Lessons from hard satisfiability problems. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 71:066707, 07 2005.
- [Battiti and Protasi, 2001] Roberto Battiti and M. Protasi. Reactive local search for the maximum clique problem1. *Algorithmica (New York)*, 29:610–637, 04 2001.

- [Bel, 2017] Belib réseau parisien de bornes de recharges pour véhicules électriques. [www.data.gouv.fr/fr/datasets/belib-reseau-parisien-de-bornes-de-recharges-accelerees-22-kw-ac-dc-pour-](http://www.data.gouv.fr/fr/datasets/belib-reseau-parisien-de-bornes-de-recharges-accelerees-22-kw-ac-dc-pour-2017) 2017.
- [Bell, 1964] J. S. Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1:195–200, Nov 1964.
- [Benioff, 1982] Paul Benioff. Quantum mechanical models of turing machines that dissipate no energy. *Phys. Rev. Lett.*, 48:1581–1585, Jun 1982.
- [Berg *et al.*, 2022] Ewout van den Berg, Zlatko K. Mineev, Abhinav Kandala, and Kristan Temme. Probabilistic error cancellation with sparse pauli-lindblad models on noisy quantum processors, 2022.
- [Bergholm *et al.*, 2018] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2018.
- [Bevern *et al.*, 2014] Renévan Bevern, Matthias Mnich, Rolf Niedermeier, and Weller Mathias. Interval scheduling and colorful independent sets. *arXiv:1402.0851v2*, 2014.
- [Bian *et al.*, 2014] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G. Macready, and Aidan Roy. Discrete optimization using quantum annealing on sparse ising models. *Frontiers in Physics*, 2, 2014.
- [Bian *et al.*, 2016] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G. Macready, and Aidan Roy. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis, 2016.
- [Bittel and Kliesch, 2021] Lennart Bittel and Martin Kliesch. Training variational quantum algorithms is NP-hard. *Physical Review Letters*, 127(12), sep 2021.
- [Boixo *et al.*, 2014] Sergio Boixo, Troels F. Rønnow, Sergei V. Isakov, Zhihui Wang, David Wecker, Daniel A. Lidar, John M. Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218–224, feb 2014.
- [Bomze *et al.*, 1999] Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers, 1999.
- [Boppana, 1990] Ravi B. Boppana. Approximating maximum independent sets by excluding subgraphs 1. 1990.
- [Brady *et al.*, 2021] Lucas T. Brady, Christopher L. Baldwin, Aniruddha Bapat, Yaroslav Kharkov, and Alexey V. Gorshkov. Optimal protocols in quantum annealing and quantum approximate optimization algorithm problems. *Physical Review Letters*, 126(7), feb 2021.
- [Braidă *et al.*, 2022] Arthur Braidă, Simon Martiel, and Ioan Todinca. On constant-time quantum annealing and guaranteed approximations for graph optimization problems, 2022.

- [Braine *et al.*, 2021] Lee Braine, Daniel J. Egger, Jennifer Glick, and Stefan Woerner. Quantum algorithms for mixed binary optimization applied to transaction settlement. *IEEE Transactions on Quantum Engineering*, 2:1–8, 2021.
- [Brandao *et al.*, 2018] Fernando G. S. L. Brandao, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For fixed control parameters the quantum approximate optimization algorithm’s objective function value concentrates for typical instances, 2018.
- [Bravyi and Gosset, 2017] Sergey Bravyi and David Gosset. Polynomial-time classical simulation of quantum ferromagnets. *Physical Review Letters*, 119(10), sep 2017.
- [Bravyi *et al.*, 2006] Sergey Bravyi, David P. DiVincenzo, Roberto I. Oliveira, and Barbara M. Terhal. The complexity of stoquastic local hamiltonian problems. 2006.
- [Bravyi *et al.*, 2020] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection. *Phys. Rev. Lett.*, 125:260505, Dec 2020.
- [Bravyi *et al.*, 2021] Sergey Bravyi, David Gosset, and Ramis Movassagh. Classical algorithms for quantum mean values. *Nature Physics*, 17(3):337–341, jan 2021.
- [Bravyi *et al.*, 2022] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Hybrid quantum-classical algorithms for approximate graph coloring. *Quantum*, 6:678, March 2022.
- [Brélaz, 1979] Daniel Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, apr 1979.
- [Breu and Kirkpatrick, 1998] Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry*, 9(1):3–24, 1998. Special Issue on Geometric Representations of Graphs.
- [Burke and Kendall, 2006] E.K. Burke and G. Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. SpringerLink: Springer e-Books. Springer US, 2006.
- [Carette and Lemonnier, 2022] Titouan Carette and Louis Lemonnier. Large-scale quantum diagrammatic reasoning tools, !-boxes vs. scalable notations, 2022.
- [Cerezo *et al.*, 2021a] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, Aug 2021.
- [Cerezo *et al.*, 2021b] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), mar 2021.
- [Cerf *et al.*, 2000] Nicolas J. Cerf, Lov K. Grover, and Colin P. Williams. Nested quantum search and structured problems. , 61(3):032303, March 2000.
- [Chan and Har-Peled, 2012] T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete and Computational Geometry*, 48(2):373–392, 2012.

- [Chiani *et al.*, 2003] Marco Chiani, Senior Member, Davide Dardari, and Marvin K. Simon. New exponential bounds and approximations for the computation of error probability in fading channels. *IEEE Transactions on Wireless Communications*, pages 840–845, 2003.
- [Childs and Wiebe, 2012] Andrew M. Childs and Nathan Wiebe. Hamiltonian Simulation Using Linear Combinations of Unitary Operations. *arXiv e-prints*, page arXiv:1202.5822, February 2012.
- [Choi, 2008] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Inform Process*, 7, 04 2008.
- [Chou *et al.*, 2021] Chi-Ning Chou, Peter J. Love, Juspreet Singh Sandhu, and Jonathan Shi. Limitations of local quantum algorithms on random max-k-xor and beyond, 2021.
- [Chuzhoy *et al.*, 2006] J. Chuzhoy, R. Ostrovsky, and Rabani Y. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.
- [CLT, 2008] *Central Limit Theorem*, pages 66–68. Springer New York, New York, NY, 2008.
- [Coecke and Duncan, 2011] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, apr 2011.
- [Coecke and Kissinger, 2017] Bob Coecke and Aleks Kissinger. *Picturing Phases and Complementarity*, pages 510–623. Cambridge University Press, 2017.
- [Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.
- [Cowtan *et al.*, 2020] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase gadget synthesis for shallow circuits. *Electronic Proceedings in Theoretical Computer Science*, 318:213–228, may 2020.
- [Cplex, 2009] IBM ILOG Cplex. V12. 1: User’s manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.
- [Crescenzi *et al.*, 2001] Pierluigi Crescenzi, Riccardo Silvestri, and Luca Trevisan. On weighted vs unweighted versions of combinatorial optimization problems. *Information and Computation*, 167(1):10–26, 2001.
- [Crooks, 2018] Gavin E. Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem, 2018.
- [Crooks, 2019] Gavin E. Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition, 2019.
- [Crosson *et al.*, 2014] Elizabeth Crosson, Edward Farhi, Cedric Yen-Yu Lin, Han-Hsuan Lin, and Peter Shor. Different strategies for optimization using the quantum adiabatic algorithm, 2014.
- [Dalyac and Henriët, 2022] Constantin Dalyac and Loïc Henriët. Embedding the mis problem for non-local graphs with bounded degree using 3d arrays of atoms, 2022.

- [Dalyac *et al.*, 2021] Constantin Dalyac, Loïc Henriët, Emmanuel Jeandel, Wolfgang Lechner, Simon Perdrix, Marc Porcheron, and Margarita Veshchezerova. 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, May 2021.
- [Dantzig and Wolfe, 1960] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.
- [de Beaudrap and Horsman, 2020] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, January 2020.
- [de Beaudrap *et al.*, 2020] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to reduce /4-parity-phase circuits, motivated by the ZX calculus. *Electronic Proceedings in Theoretical Computer Science*, 318:131–149, may 2020.
- [de Beaudrap *et al.*, 2021] Niel de Beaudrap, Aleks Kissinger, and Konstantinos Meichanetzidis. Tensor network rewriting strategies for satisfiability and counting. *Electronic Proceedings in Theoretical Computer Science*, 340:46–59, sep 2021.
- [de la Grand’rive and Hullo, 2019] Pierre Dupuy de la Grand’rive and Jean-Francois Hullo. Knapsack problem variants of qaoa for battery revenue optimisation, 2019.
- [de Wolf, 2019] Ronald de Wolf. Quantum computing: Lecture notes, 2019.
- [Desrosiers and Lübbecke, 2006] Jacques Desrosiers and Marco Lübbecke. *A Primer in Column Generation*, pages 1–32. 03 2006.
- [Deutsch and Jozsa, 1992] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439:553 – 558, 1992.
- [Deutsch, 1985] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400:117 – 97, 1985.
- [Deza and Laurent, 1994] Michel Deza and Monique Laurent. Applications of cut polyhedra — ii. *Journal of Computational and Applied Mathematics*, 55(2):217–247, 1994.
- [Dial, 2022] Oliver Dial. Eagle’s quantum performance progress, Mar 2022.
- [Doerr *et al.*, 2011] Benjamin Doerr, Anton Eremeev, Frank Neumann, Madeleine Theile, and Christian Thyssen. Evolutionary algorithms and dynamic programming. *Theoretical Computer Science*, 412(43):6020–6035, oct 2011.
- [Duncan and Perdrix, 2009] Ross Duncan and Simon Perdrix. Graph states and the necessity of euler decomposition. In Klaus Ambos-Spies, Benedikt Löwe, and Wolfgang Merkle, editors, *Mathematical Theory and Computational Practice*, pages 167–177, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [Duncan and Perdrix, 2010] Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In Samson Abramsky, Cyril Gavioille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 285–296, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

- [Duncan and Perdrix, 2014] Ross Duncan and Simon Perdrix. Pivoting makes the zx-calculus complete for real stabilizers. *Electronic Proceedings in Theoretical Computer Science*, 171:50–62, Dec 2014.
- [Duncan *et al.*, 2020] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum*, 4:279, jun 2020.
- [Dupret *et al.*, 2021] Muriel Dupret, Jean-Paul Zimmermann, Nicolas Andreau, and Mickael Guernevel. Panel usages électrodomestiques - consommations électrodomestiques françaises basées sur les mesures collectées en continu dans 100 logements, March 2021.
- [East *et al.*, 2022] Richard D.P. East, John van de Wetering, Nicholas Chancellor, and Adolfo G. Grushin. AKLT-states as ZX-diagrams: Diagrammatic reasoning for quantum states. *PRX Quantum*, 3(1), jan 2022.
- [Egger *et al.*, 2021] Daniel J. Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, June 2021.
- [Eppstein, 1992] David Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.
- [Ezratty, 2021] Olivier Ezratty. Understanding quantum technologies, 2021.
- [Farhi and Harrow, 2016] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm, 2016.
- [Farhi *et al.*, 2000] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution, 2000.
- [Farhi *et al.*, 2002] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing, 2002.
- [Farhi *et al.*, 2014a] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.
- [Farhi *et al.*, 2014b] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, 2014.
- [Farhi *et al.*, 2020] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: A typical case, 2020.
- [Farhi *et al.*, 2022] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou. The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size. *Quantum*, 6:759, jul 2022.
- [Feo *et al.*, 1994] Thomas A. Feo, Mauricio G. C. Resende, and Stuart H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.
- [Feynman, 1986] Richard P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, Jun 1986.

- [Fletcher, 1987] Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, NY, USA, second edition, 1987.
- [Fontana *et al.*, 2020] Enrico Fontana, M. Cerezo, Andrew Arrasmith, Ivan Rungger, and Patrick J. Coles. Optimizing parametrized quantum circuits via noise-induced breaking of symmetries, 2020.
- [Ford and Fulkerson, 1958] L. R. Ford and D. R. Fulkerson. *A suggested computation for maximal multi-commodity network flows*. RAND Corporation, Santa Monica, CA, 1958.
- [França and García-Patrón, 2021] Daniel Stilck França and Raul García-Patrón. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics*, 17(11):1221–1227, oct 2021.
- [Frieze and Jerrum, 1997] A. Frieze and M. Jerrum. Improved approximation algorithms for max-k-cut and max bisection. *Algorithmica*, 18(1):67–81, May 1997.
- [Fuchs *et al.*, 2020] Franz Georg Fuchs, Herman Øie Kolden, Niels Henrik Aase, and Giorgio Sartor. Efficient encoding of the weighted max k-cut on a quantum computer using qaoa, 2020.
- [Gao and Han, 2012] Fuchang Gao and Lixing Han. Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51:259–277, 05 2012.
- [Garey and Johnson, 1976] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, jan 1976.
- [Garey and Johnson, 1978] M. R. Garey and D. S. Johnson. “strong” np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, jul 1978.
- [Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979.
- [Garey and Johnson, 1990] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman Co., USA, 1990.
- [Garvie and Duncan, 2018] Liam Garvie and Ross Duncan. Verifying the smallest interesting colour code with quantomatic. *Electronic Proceedings in Theoretical Computer Science*, 266:147–163, feb 2018.
- [Gass and Harris, 2001] Saul I. Gass and Carl M. Harris. *Strong duality theorem* *Strong duality theorem*, pages 790–790. Springer US, New York, NY, 2001.
- [Gavril, 1972] Fanica Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM J. Comput.*, 1:180–187, 1972.
- [Geman and Geman, 1984] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

- [Geoffrion, 1974] A. M. Geoffrion. *Lagrangian relaxation for integer programming*, pages 82–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 1974.
- [Gibbons *et al.*, 1993] Luana E. Gibbons, Donald W. Hearn, and Panos M. Pardalos. A continuous based heuristic for the maximum clique problem. In *Cliques, Coloring, and Satisfiability*, 1993.
- [Gilmore and Gomory, 1963] P.C. Gilmore and Ralph Gomory. A linear programming approach to the cutting stock problem—part ii. *Operations Research*, 11, 12 1963.
- [Glos *et al.*, 2022] Adam Glos, Aleksandra Krawiec, and Zoltán Zimborás. Space-efficient binary optimization for variational quantum computing. *npj Quantum Information*, 8:39, 04 2022.
- [Goemans and Williamson, 1995] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995.
- [Graham *et al.*, 1979] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. Elsevier, 1979.
- [Grossi *et al.*, 2006] Giuliano Grossi, Massimo Marchi, and Roberto Posenato. Solving maximum independent set by asynchronous distributed hopfield-type neural networks. *RAIRO - Theoretical Informatics and Applications*, 40:371–388, 04 2006.
- [Grover, 1996] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv e-prints*, pages quant-ph/9605043, May 1996.
- [Grötschel *et al.*, 1984] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. In C. Berge and V. Chvátal, editors, *Topics on Perfect Graphs*, volume 88 of *North-Holland Mathematics Studies*, pages 325–356. North-Holland, 1984.
- [G.S L. Brandão *et al.*, 2022] Fernando G.S L. Brandão, Richard Kueng, and Daniel Stilck França. Faster quantum and classical SDP approximations for quadratic binary optimization. *Quantum*, 6:625, January 2022.
- [Guerreschi and Matsuura, 2019] G. G. Guerreschi and A. Y. Matsuura. QAOA for max-cut requires hundreds of qubits for quantum speed-up. *Scientific Reports*, 9(1), may 2019.
- [Guerreschi and Smelyanskiy, 2017] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms, 2017.
- [Gurobi Optimization, LLC, 2022] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.
- [Gärtner and Matoušek, 2013] Bernd Gärtner and Jiří Matoušek. *Approximation Algorithms and Semidefinite Programming*. 11 2013.
- [Haapasalo, 2019] Erkki Haapasalo. The choi-jamiołkowski isomorphism and covariant quantum channels, 2019.

- [Habib *et al.*, 2000] Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1):59–84, 2000.
- [Hadfield *et al.*, 2019] Stuart Hadfield, Zihui Wang, Bryan O’Gorman, Eleanor Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, feb 2019.
- [Hadfield *et al.*, 2021] Stuart Hadfield, Tad Hogg, and Eleanor Rieffel. Analytical framework for quantum alternating operator ansätze, 05 2021.
- [Hadfield, 2018] Stuart Hadfield. Quantum algorithms for scientific computing and approximate optimization, 2018.
- [Hadfield, 2021] Stuart Hadfield. On the representation of boolean and real functions as hamiltonians for quantum computing. *ACM Transactions on Quantum Computing*, 2(4):1–21, Dec 2021.
- [Hadzahasanovic *et al.*, 2018] Amar Hadzahasanovic, Kang Feng Ng, and Quanlong Wang. Two complete axiomatisations of pure-state qubit quantum computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS ’18*, pages 502–511, New York, NY, USA, 2018. ACM.
- [Halldórsson and Karlsson, 2006] Magnús M. Halldórsson and Ragnar K. Karlsson. Strip graphs: Recognition and scheduling. In Fedor V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science*, pages 137–146, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [Halldórsson, 1993] Magnús M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, 45(1):19–23, 1993.
- [Halperin *et al.*, 2004] Eran Halperin, Dror Livnat, and Uri Zwick. Max cut in cubic graphs. *Journal of Algorithms*, 53(2):169–185, 2004.
- [Hanks *et al.*, 2020] Michael Hanks, Marta P. Estarellas, William J. Munro, and Kae Nemoto. Effective compression of quantum braided circuits aided by ZX-calculus. *Physical Review X*, 10(4), nov 2020.
- [Harrigan *et al.*, 2021] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Daniel Eppens, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Alan Ho, Sabrina Hong, Trent Huang, L. B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Martin Leib, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mroczkiewicz, Josh Mutus, Ofer Naaman, Charles Neill, Florian Neukart, Murphy Yuezhen Niu, Thomas E. O’Brien, Bryan O’Gorman, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Andrea Skolik, Vadim Smelyanskiy, Doug Strain, Michael Streif, Marco Szalay, Amit Vainsencher, Theodore White, Z. Jamie Yao, Ping

- Yeh, Adam Zalcman, Leo Zhou, Hartmut Neven, Dave Bacon, Erik Lucero, Edward Farhi, and Ryan Babbush. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, feb 2021.
- [Håstad, 1999] Johan Håstad. Clique is hard to approximate within $n^{1/4}$. *Acta Mathematica*, 182(1):105 – 142, 1999.
- [Hastings and Freedman, 2013] M. B. Hastings and M. H. Freedman. Obstructions to classically simulating the quantum adiabatic algorithm. 2013.
- [Hastings, 2019] M. B. Hastings. Classical and quantum bounded depth approximation algorithms, 2019.
- [Held *et al.*, 2012] Stephan Held, William Cook, and Edward Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, 4, 12 2012.
- [Hen and Sarandy, 2016] Itay Hen and Marcelo S. Sarandy. Driver hamiltonians for constrained optimization in quantum annealing. *Physical Review A*, 93(6), jun 2016.
- [Hen and Spedalieri, 2016] Itay Hen and Federico M. Spedalieri. Quantum annealing for constrained optimization. *Phys. Rev. Applied*, 5:034007, Mar 2016.
- [Henriet *et al.*, 2020] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020.
- [Hoffman and Padberg, 1985] K. Hoffman and M. Padberg. Lp-based combinatorial problem solving. *Annals of Operations Research*, 4(1):145–194, Dec 1985.
- [Homer and Peinado, 1994] Steven Homer and Marcus Peinado. On the performance of polynomial-time clique approximation algorithms on very large graphs. 1994.
- [Horsman, 2011] Clare Horsman. Quantum pictorialism for topological cluster-state computing. *New Journal of Physics*, 13(9):095011, sep 2011.
- [Husfeldt, 2015] Thore Husfeldt. Graph colouring algorithms, 2015.
- [Huyer and Neumaier, 2008] Waltraud Huyer and Arnold Neumaier. Snobfit - stable noisy optimization by branch and fit. *ACM Trans. Math. Softw.*, 35, 07 2008.
- [Ibarra and Kim, 1975] Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22:463–468, 1975.
- [Ising, 1925] Ernst Ising. Contribution to the Theory of Ferromagnetism. *Z. Phys.*, 31:253–258, 1925.
- [Izmaylov *et al.*, 2021] Artur F. Izmaylov, Robert A. Lang, and Tzu-Ching Yen. Analytic gradients in variational quantum algorithms: Algebraic extensions of the parameter-shift rule to general unitary transformations. *Physical Review A*, 104(6), dec 2021.
- [Jansen *et al.*, 2007] Sabine Jansen, Mary-Beth Ruskai, and Ruedi Seiler. Bounds for the adiabatic approximation with applications to quantum computation. *Journal of Mathematical Physics*, 48(10):102111, 2007.

- [Jeandel *et al.*, 2018a] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics. In *The 33rd Annual {ACM/IEEE} Symposium on Logic in Computer Science, {LICS} 2018*, Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, pages 559–568, Oxford, United Kingdom, July 2018.
- [Jeandel *et al.*, 2018b] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Diagrammatic Reasoning beyond Clifford+T Quantum Mechanics. In *The 33rd Annual Symposium on Logic in Computer Science*, Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, pages 569–578, Oxford, United Kingdom, July 2018.
- [Jeandel *et al.*, 2019] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A generic normal form for zx-diagrams and application to the rational angle completeness. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '19. IEEE Press, 2019.
- [Jeandel *et al.*, 2022] Emmanuel Jeandel, Simon Perdrix, and Margarita Veshchezerova. Addition and Differentiation of ZX-diagrams. *arXiv e-prints*, page arXiv:2202.11386, February 2022.
- [Johnson and Trick, 1996] David J. Johnson and Michael A. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, USA, 1996.
- [Johnson, 1974] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.
- [Jozsa, 2005] Richard Jozsa. An introduction to measurement based quantum computation, 2005.
- [Kadowaki and Nishimori, 1998] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355–5363, nov 1998.
- [Kandala *et al.*, 2019] Abhinav Kandala, Kristan Temme, Antonio D. Córcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567(7749):491–495, Mar 2019.
- [Kann *et al.*, 1997] Viggo Kann, Sanjeev Khanna, Jens Lagergren, and Alessandro Panconesi. On the hardness of approximating max k-cut and its dual. *Chicago Journal of Theoretical Computer Science*, 2, 1997.
- [Karloff, 1999] Howard Karloff. How good is the goemans-williamson max cut algorithm? *SIAM Journal on Computing*, 29(1):336–350, 1999.
- [Karmarkar, 1984] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, page 302–311, New York, NY, USA, 1984. Association for Computing Machinery.
- [Karp, 1972] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.
- [Keller and Vanden-Broeck, 2007] Joseph B. Keller and Jean-Marc Vanden-Broeck. Stirling's formula derived simply, 2007.

- [Kelley, 2011] Carl Kelley. Implicit filtering. 01 2011.
- [Kempe *et al.*, 2006] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006.
- [Kempe *et al.*, 2007] Julia Kempe, Oded Regev, and Ben Toner. Unique games with entangled provers are easy, 2007.
- [Khachiyan, 1980] L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.
- [Khairy *et al.*, 2020] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash. Learning to optimize variational quantum circuits to solve combinatorial problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2367–2375, apr 2020.
- [Khot *et al.*, 2007] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O’Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007.
- [Khot, 2002] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC ’02, page 767–775, New York, NY, USA, 2002. Association for Computing Machinery.
- [King *et al.*, 2015] James King, Sheir Yarkoni, Mayssam M. Nevisi, Jeremy P. Hilton, and Catherine C. McGeoch. Benchmarking a quantum annealing processor with the time-to-target metric, 2015.
- [Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.
- [Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [Kissinger and van de Wetering, 2019] Aleks Kissinger and John van de Wetering. Universal MBQC with generalised parity-phase interactions and pauli measurements. *Quantum*, 3:134, apr 2019.
- [Kissinger and van de Wetering, 2020a] Aleks Kissinger and John van de Wetering. PyZX: Large scale automated diagrammatic reasoning. *Electronic Proceedings in Theoretical Computer Science*, 318:229–241, may 2020.
- [Kissinger and van de Wetering, 2020b] Aleks Kissinger and John van de Wetering. Reducing the number of non-clifford gates in quantum circuits. *Physical Review A*, 102(2), aug 2020.
- [Kissinger *et al.*, 2014] Aleks Kissinger, Alex Merry, and Matvey Soloviev. Pattern graph rewrite systems. *Electronic Proceedings in Theoretical Computer Science*, 143:54–66, mar 2014.
- [Kiukas *et al.*, 2019] Jukka Kiukas, Pekka Lahti, Juha-Pekka Pellonpää, and Kari Ylinen. Complementary observables in quantum mechanics. *Foundations of Physics*, 49(6):506–531, Jun 2019.

- [Kochenberger *et al.*, 2005] Gary Kochenberger, Fred Glover, Bahram Alidaee, and Cesar Rego. An unconstrained quadratic binary programming approach to the vertex coloring problem. *Annals OR*, 139:229–241, 10 2005.
- [Kochenberger *et al.*, 2013] Gary A. Kochenberger, Jin-Kao Hao, Zhipeng Lü, Haibo Wang, and Fred Glover. Solving large scale max cut problems via tabu search. *Journal of Heuristics*, 19(4):565–571, Aug 2013.
- [Kochenberger *et al.*, 2014] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, July 2014.
- [Koczor and Benjamin, 2022] Bálint Koczor and Simon C. Benjamin. Quantum analytic descent. *Physical Review Research*, 4(2), apr 2022.
- [Kolen *et al.*, 2007] Antoon W.J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C.R. Spieksma. Interval scheduling: A survey. www.interscience.wiley.com, 2007.
- [Kuijpers *et al.*, 2019] Stach Kuijpers, John van de Wetering, and Aleks Kissinger. Graphical fourier theory and the cost of quantum addition, 2019.
- [Kyriienko and Elfving, 2021] Oleksandr Kyriienko and Vincent E. Elfving. Generalized quantum circuit differentiation rules. *Phys. Rev. A*, 104:052417, Nov 2021.
- [Larkin *et al.*, 2020] Jason Larkin, Matías Jonsson, Daniel Justice, and Gian Giacomo Guerreschi. Evaluation of qaoa based on the approximation ratio of individual samples, 2020.
- [Lavrijsen *et al.*, 2020] Wim Lavrijsen, Ana Tudor, Juliane Muller, Costin Iancu, and Wibe de Jong. Classical optimizers for noisy intermediate-scale quantum devices. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, oct 2020.
- [Lawler, 1976] Eugene L. Lawler. A note on the complexity of the chromatic number problem. *Inf. Process. Lett.*, 5:66–67, 1976.
- [Le Digabel, 2011] Sébastien Le Digabel. Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm. 37(4), feb 2011.
- [Lewis, 2021] R. M. R. Lewis. *Applications and Extensions*, pages 155–202. Springer International Publishing, Cham, 2021.
- [Lloyd, 2018] Seth Lloyd. Quantum approximate optimization is computationally universal, 2018.
- [Lotshaw *et al.*, 2022] Phillip C. Lotshaw, Thien Nguyen, Anthony Santana, Alexander McCaskey, Rebekah Herrman, James Ostrowski, George Siopsis, and Travis S. Humble. Scaling quantum approximate optimization on near-term hardware, 2022.
- [Lovász, 1979] László Miklós Lovász. On the shannon capacity of a graph. *IEEE Trans. Inf. Theory*, 25:1–7, 1979.
- [Lu *et al.*, 2017] Can Lu, Jeffrey Xu Yu, Hao Wei, and Yikai Zhang. Finding the maximum clique in massive graphs. *Proc. VLDB Endow.*, 10(11):1538–1549, aug 2017.

- [Lucas, 2014] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, 2014.
- [Manin, 1980] Yuri Manin. *Vichislimoe i nevichislimoe (Computable and uncomputable)*. soviet radio, 1980.
- [Marchiori, 1998] Elena Marchiori. A simple heuristic based genetic algorithm for the maximum clique problem. In *Proceedings of the 1998 ACM Symposium on Applied Computing*, SAC '98, page 366–373, New York, NY, USA, 1998. Association for Computing Machinery.
- [Mari *et al.*, 2021] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. Estimating the gradient and higher-order derivatives on quantum hardware. *Physical Review A*, 103(1), jan 2021.
- [Martin-Mayor and Hen, 2015] Victor Martin-Mayor and Itay Hen. Unraveling quantum annealers using classical hardness. *Scientific Reports*, 5(1), oct 2015.
- [Martoňák *et al.*, 2002] Roman Martoňák, Giuseppe E. Santoro, and Erio Tosatti. Quantum annealing by the path-integral monte carlo method: The two-dimensional random ising model. *Phys. Rev. B*, 66:094203, Sep 2002.
- [Martoňák *et al.*, 2004] Roman Martoňák, Giuseppe E. Santoro, and Erio Tosatti. Quantum annealing of the traveling-salesman problem. *Phys. Rev. E*, 70:057701, Nov 2004.
- [Marwaha, 2021] Kunal Marwaha. Local classical MAX-CUT algorithm outperforms mml:math xmlns:mml="http://www.w3.org/1998/math/MathML" mml:mip/mml:mimml:mo=/mml:momml:mn2/mml:m mml:mo=" " mml:mi="QAOA" mml:mi="on" mml:mi="high-girth" mml:mi="regular" mml:mi="graphs." *Quantum*, 5:437, apr 2021.
- [Marx, 2003] Dániel Marx. Graph colouring problems and their applications in scheduling, 2003.
- [Matula, 1976] David W. Matula. The largest clique size in a random graph. Technical report, Department of Computer Science, Southern Methodist University, 1976.
- [Mauri and Lorena, 2012] Geraldo Regis Mauri and Luiz Antonio Nogueira Lorena. A column generation approach for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, 217(1):69–74, 2012.
- [Mbeng *et al.*, 2019a] Glen Bigan Mbeng, Rosario Fazio, and Giuseppe Santoro. Quantum annealing: a journey through digitalization, control, and hybrid quantum variational schemes, 2019.
- [Mbeng *et al.*, 2019b] Glen Bigan Mbeng, Rosario Fazio, and Giuseppe E. Santoro. Optimal quantum control with digitized quantum annealing, 2019.
- [McClean *et al.*, 2018] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), nov 2018.
- [McClean *et al.*, 2021] Jarrod R. McClean, Matthew P. Harrigan, Masoud Mohseni, Nicholas C. Rubin, Zhang Jiang, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Low-depth mechanisms for quantum optimization. *PRX Quantum*, 2(3), jul 2021.
- [Mehrotra and Trick, 1996] Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8(4):344–354, 1996.

- [Mitarai *et al.*, 2018] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3), sep 2018.
- [Montanaro, 2020] Ashley Montanaro. Quantum speedup of branch-and-bound algorithms. *Phys. Rev. Research*, 2:013056, Jan 2020.
- [Moosavian *et al.*, 2022] Ali Hamed Moosavian, Seyed Sajad Kahani, and Salman Beigi. Limits of Short-Time Evolution of Local Hamiltonians. *Quantum*, 6:744, June 2022.
- [Morita and Nishimori, 2008] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12):125210, dec 2008.
- [Motzkin and Straus, 1965] Theodore S. Motzkin and Ernst G. Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian Journal of Mathematics*, 17:533 – 540, 1965.
- [Moussa *et al.*, 2020] Charles Moussa, Henri Calandra, and Vedran Dunjko. To quantum or not to quantum: towards algorithm selection in near-term quantum optimization. *Quantum Science and Technology*, 5(4):044009, oct 2020.
- [Moussa *et al.*, 2022] Charles Moussa, Hao Wang, Thomas Bäck, and Vedran Dunjko. Unsupervised strategies for identifying optimal parameters in quantum approximate optimization algorithm. *EPJ Quantum Technology*, 9(1):11, May 2022.
- [Nadarajah and Kotz, 2008] Saralees Nadarajah and Samuel Kotz. Exact distribution of the max/min of two gaussian random variables. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(2):210–212, 2008.
- [Nash, 2000] John C. Nash. The (dantzig) simplex method for linear programming. In *Computing in Science and Engg.*, volume 2, pages 29–31. IEEE Educational Activities Department, Piscataway, NJ, USA, 2000.
- [Neven *et al.*, 2008] Hartmut Neven, Geordie Rose, and William G. Macready. Image recognition with an adiabatic quantum computer i. mapping to quadratic unconstrained binary optimization, 2008.
- [Ng and Wang, 2018] Kang Feng Ng and Quanlong Wang. Completeness of the zx-calculus for pure qubit clifford+t quantum mechanics, 2018.
- [Nieberg *et al.*, 2004] T. Nieberg, J. Hurink, and Walter K. A robust ptas for maximum weightindependent sets in unit disk graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 214,221, 2004.
- [Nielsen and Chuang, 2011] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.
- [Ohzeki, 2020] Masayuki Ohzeki. Breaking limitation of quantum annealer in solving optimization problems under constraints, 2020.
- [Pagano *et al.*, 2020] Guido Pagano, Aniruddha Bapat, Patrick Becker, Katherine S. Collins, Arinjoy De, Paul W. Hess, Harvey B. Kaplan, Antonis Kyprianidis, Wen Lin Tan, Christopher Baldwin, Lucas T. Brady, Abhinav Deshpande, Fangli Liu, Stephen Jordan, Alexey V. Gorshkov, and Christopher Monroe. Quantum approximate optimization of the long-range

- ising model with a trapped-ion quantum simulator. *Proceedings of the National Academy of Sciences*, 117(41):25396–25401, oct 2020.
- [Pardalos and Jha, 1992] Panos M Pardalos and Somesh Jha. Complexity of uniqueness and local search in quadratic 0–1 programming. *Operations Research Letters*, 11(2):119–123, 1992.
- [Park and Boyd, 2017] Jaehyun Park and Stephen Boyd. General heuristics for nonconvex quadratically constrained quadratic programming, 2017.
- [Patel *et al.*, 2022] Yash J. Patel, Sofiene Jerbi, Thomas Bäck, and Vedran Dunjko. Reinforcement learning assisted recursive qaoa, 2022.
- [Perdrix and Wang, 2016] Simon Perdrix and Quanlong Wang. Supremacy is Necessary for Quantum Diagram Reasoning. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76:1–76:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [Peruzzo *et al.*, 2014] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), jul 2014.
- [Pfeuty, 1970] Pierre Pfeuty. The one-dimensional ising model with a transverse field. *Annals of Physics*, 57(1):79–90, 1970.
- [Pichler *et al.*, 2018] Hannes Pichler, Sheng-Tao Wang, Leo Zhou, Soonwon Choi, and Mikhail D. Lukin. Quantum optimization for maximum independent set using rydberg atom arrays, 2018.
- [Powell, 1994] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. 1994.
- [Powell, 2009] M. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Technical Report, Department of Applied Mathematics and Theoretical Physics*, 01 2009.
- [Preskill, 2018] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, aug 2018.
- [Qiang *et al.*, 2018] Xiaogang Qiang, Xiaoqi Zhou, Jianwei Wang, Callum M. Wilkes, Thomas Loke, Sean O’Gara, Laurent Kling, Graham D. Marshall, Raffaele Santagati, Timothy C. Ralph, Jingbo B. Wang, Jeremy L. O’Brien, Mark G. Thompson, and Jonathan C. F. Matthews. Large-scale silicon quantum photonics implementing arbitrary two-qubit processing. *Nature Photonics*, 12(9):534–539, aug 2018.
- [QLM, 2022] Quantum learning machine, 2022.
- [Reichardt, 2004] Ben W. Reichardt. The quantum adiabatic optimization algorithm and local minima. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, STOC ’04*, page 502–510, New York, NY, USA, 2004. Association for Computing Machinery.
- [Rieffel *et al.*, 2014] Eleanor G. Rieffel, Davide Venturelli, Bryan O’Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14(1):1–36, dec 2014.

- [Ryan and Foster, 1981] David Ryan and Brian Foster. An integer programming approach to scheduling. *Computer Scheduling of Public Transport*, 1:269–, 01 1981.
- [Rønnow *et al.*, 2014] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014.
- [Sack and Serbyn, 2021] Stefan H. Sack and Maksym Serbyn. Quantum annealing initialization of the quantum approximate optimization algorithm. *Quantum*, 5:491, July 2021.
- [Sahni, 1976] Sartaj K. Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, jan 1976.
- [Sanders *et al.*, 2020] Yuval R. Sanders, Dominic W. Berry, Pedro C.S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. Compilation of fault-tolerant quantum heuristics for combinatorial optimization. *PRX Quantum*, 1(2), nov 2020.
- [Schuld *et al.*, 2019] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), mar 2019.
- [Shaydulin and Alexeev, 2019] Ruslan Shaydulin and Yuri Alexeev. Evaluating quantum approximate optimization algorithm: A case study. In *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*, pages 1–6, 2019.
- [Shaydulin *et al.*, 2021] Ruslan Shaydulin, Kunal Marwaha, Jonathan Wurtz, and Phillip C. Lotshaw. Qaoakit: A toolkit for reproducible study, application, and verification of the qaoa. In *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, pages 64–71, 2021.
- [Shaydulin *et al.*, 2022] Ruslan Shaydulin, Phillip C. Lotshaw, Jeffrey Larson, James Ostrowski, and Travis S. Humble. Parameter transfer for quantum approximate optimization of weighted maxcut, 2022.
- [Sherrington and Kirkpatrick, 1975] David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Phys. Rev. Lett.*, 35:1792–1796, Dec 1975.
- [Shin *et al.*, 2014] Seung Woo Shin, Graeme Smith, John A. Smolin, and Umesh Vazirani. How "quantum" is the d-wave machine?, 2014.
- [Shor, 1995] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *arXiv e-prints*, pages quant-ph/9508027, August 1995.
- [Simon, 1997] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.
- [Simone *et al.*, 1995] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of ising spin glasses: New experimental results with a branch-and-cut algorithm, 1995.
- [Skutella and Woeginger, 2000] Martin Skutella and Gerhard J. Woeginger. A ptas for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research*, 25(1):63–75, 2000.

- [Skutella, 1998] Martin Skutella. Semidefinite relaxations for parallel machine scheduling. Technical report, Technische Universität Berlin, 1998.
- [Smith, 1956] W. E. Smith. Various optimizers for single-stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1956.
- [Spieksma, 1999] Frits C. R. Spieksma. On the approximability of an interval scheduling problem. *J. of Scheduling*, 2(5):215–227, 1999.
- [Stollenwerk and Hadfield, 2022] Tobias Stollenwerk and Stuart Hadfield. Diagrammatic Analysis for Parameterized Quantum Circuits. *arXiv e-prints*, page arXiv:2204.01307, April 2022.
- [Stollenwerk *et al.*, 2018] Tobias Stollenwerk, Elisabeth Lobe, and Martin Jung. Flight gate assignment with a quantum annealer, 2018.
- [Stone, 1932] M. H. Stone. On one-parameter unitary groups in hilbert space. *Annals of Mathematics*, 33(3):643–648, 1932.
- [Svensson *et al.*, 2021] Marika Svensson, Martin Andersson, Mattias Gronkvist, Pontus Vikstaal, Devdatt P. Dubhashi, G. Ferrini, and Göran Johansson. A hybrid quantum-classical heuristic to solve large-scale integer linear programs. 2021.
- [Systems,] DWave Systems. Wave qpu architecture: Topologies.
- [Systems, 2022] D-Wave Systems. *QPU-Specific Physical Properties: Advantage system 6.1*. D-Wave Systems Inc., 05 2022.
- [Tate *et al.*, 2020] Reuben Tate, Majid Farhadi, Creston Herold, Greg Mohler, and Swati Gupta. Bridging classical and quantum with sdp initialized warm-starts for qaoa, 2020.
- [Toumi *et al.*, 2021] Alexis Toumi, Richie Yeung, and Giovanni de Felice. Diagrammatic differentiation for quantum machine learning. *arXiv e-prints*, page arXiv:2103.07960, March 2021.
- [Tran *et al.*, 2019] Minh C. Tran, Andrew Y. Guo, Yuan Su, James R. Garrison, Zachary Eldredge, Michael Foss-Feig, Andrew M. Childs, and Alexey V. Gorshkov. Locality and digital quantum simulation of power-law interactions. *Physical Review X*, 9(3), jul 2019.
- [Trevisan, 2001] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 453–461, New York, NY, USA, 2001. Association for Computing Machinery.
- [Urban *et al.*, 2009] E. Urban, T. A. Johnson, T. Henage, L. Isenhower, D. D. Yavuz, T. G. Walker, and M. Saffman. Observation of rydberg blockade between two atoms. *Nature Physics*, 5(2):110–114, jan 2009.
- [van Apeldoorn *et al.*, 2020] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-Solvers: Better upper and lower bounds. *Quantum*, 4:230, February 2020.
- [van Dam *et al.*, 2001] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001.

- [van de Wetering, 2020] John van de Wetering. Zx-calculus for the working quantum computer scientist, 2020.
- [Vanderbeck, 1994] F. Vanderbeck. *Decomposition and Column Generation for Integer Programs*. Université catholique de Louvain, 1994.
- [Vanderbei, 2014] Robert J. Vanderbei. *The Simplex Method*, pages 11–23. Springer US, Boston, MA, 2014.
- [Venturelli *et al.*, 2015] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O’Gorman, Rupak Biswas, and Vadim Smelyanskiy. Quantum optimization of fully connected spin glasses. *Physical Review X*, 5(3), sep 2015.
- [Vilmart, 2019] Renaud Vilmart. A near-optimal axiomatisation of ZX-calculus for pure qubit quantum mechanics. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019.
- [Vyskocil and Djidjev, 2019] Tomas Vyskocil and Hristo Djidjev. Embedding equality constraints of optimization problems into a quantum annealer. *Algorithms*, 12:77, 04 2019.
- [Wang and Yeung, 2021] Quanlong Wang and Richie Yeung. Representing and implementing matrices using algebraic zx-calculus, 2021.
- [Wang and Yeung, 2022] Quanlong Wang and Richie Yeung. Differentiating and integrating zx diagrams, 2022.
- [Wang *et al.*, 2017] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor Rieffel. The quantum approximation optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97, 06 2017.
- [Wang *et al.*, 2020] Zhihui Wang, Nicholas C. Rubin, Jason M. Dominy, and Eleanor G. Rieffel. *xy* mixers: Analytical and numerical results for the quantum alternating operator ansatz. *Phys. Rev. A*, 101:012320, Jan 2020.
- [Wang, 2020] Quanlong Wang. Algebraic complete axiomatisation of zx-calculus with a normal form via elementary matrix operations, 2020.
- [Wierichs *et al.*, 2022] David Wierichs, Josh Izaac, Cody Wang, and Cedric Yen-Yu Lin. General parameter-shift rules for quantum gradients. *Quantum*, 6:677, March 2022.
- [Wikipedia, 2022a] Wikipedia. Ising model — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Ising%20model&oldid=1095449468>, 2022. [Online; accessed 21-July-2022].
- [Wikipedia, 2022b] Wikipedia. Moore’s law — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Moore’s%20law&oldid=1106938703>, 2022. [Online; accessed 12-September-2022].
- [Wilson *et al.*, 2021] Max Wilson, Rachel Stromswold, Filip Wudarski, Stuart Hadfield, Norm Tubman, and Eleanor Rieffel. Optimizing quantum heuristics with meta-learning. *Quantum Machine Intelligence*, 3, 06 2021.

- [Woeginger, 2001] Gerhard Woeginger. When does a dynamic programming formulation guarantee the existence of an fptas? *Electronic Colloquium on Computational Complexity (ECCC)*, 12, 01 2001.
- [Wurtz and Love, 2021] Jonathan Wurtz and Peter Love. MaxCut quantum approximate optimization algorithm performance guarantees for `mml:math xmlns:mml="http://www.w3.org/1998/math/MathML" mml:mrow mml:mip/mml:mimml:mo>/mml:momml:Physical Review A`, 103(4), apr 2021.
- [Yang *et al.*, 2017] Zhi-Cheng Yang, Armin Rahmani, Alireza Shabani, Hartmut Neven, and Claudio Chamon. Optimizing variational quantum algorithms using pontryagin’s minimum principle. *Physical Review X*, 7(2), may 2017.
- [Yeung *et al.*,] Richie Yeung, Quanlong Wang, and Razin A. Shaikh. *How to sum and exponentiate Hamiltonians in ZXW calculus*.
- [Yonaga *et al.*, 2020] Kouki Yonaga, Masamichi J. Miyama, and Masayuki Ohzeki. Solving inequality-constrained binary optimization problems on quantum annealer, 2020.
- [Yu and Nabil, 2021] Sizhuo Yu and Tahar Nabil. Applying the hubbard-stratonovich transformation to solve scheduling problems under inequality constraints with quantum annealing. *Frontiers in Physics*, 9, 2021.
- [Zarinelli, 2012] Elia Zarinelli. *Spin-glass models and interdisciplinary applications*. Theses, Université Paris Sud - Paris XI, January 2012.
- [Zhao and Gao, 2021] Chen Zhao and Xiao-Shan Gao. Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus. *Quantum*, 5:466, June 2021.
- [Zhou *et al.*, 2020] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, Jun 2020.
- [Zhu *et al.*, 2016] Zheng Zhu, Andrew J. Ochoa, Stefan Schnabel, Firas Hamze, and Helmut G. Katzgraber. Best-case performance of quantum annealers on native spin-glass benchmarks: How chaos can affect success probabilities. *Phys. Rev. A*, 93:012317, Jan 2016.
- [Zuckerman, 2007] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007.
- [Östergård, 1999] Patric R.J. Östergård. A new algorithm for the maximum-weight clique problem. *Electronic Notes in Discrete Mathematics*, 3:153–156, 1999. 6th Twente Workshop on Graphs and Combinatorial Optimization.

Abstract

The domain of energy management involves many combinatorial optimization problems known to be computationally hard. The emergence of quantum computers suggests new approaches for these problems. For near-future machines particularly promising are variational quantum heuristics such as QAOA that can leverage the computational power of the imperfect quantum hardware.

We explore the potential of variational quantum algorithms for optimization problems issued from the field of "*smart charging*" of electrical vehicles. We consider two problems inspired by real-world usecases. In the first problem, modeled as *Max-K-Cut*, we search to schedule a set of prioritized charges on several stations while minimizing the weighted completion time. In the second problem, modeled as *Maximum Independent Set*, we aim to maximize the number of satisfied charge demands on a single station while respecting the conflicts between demands. For both problems we develop an experimental protocol specifying the *encoding step* and the *parameter optimization routine*. Our numerical experiments confirm the interest of quantum heuristics for these problems as well as the quality of our experimental protocol.

In order to extend the applicability of quantum heuristics we introduce a new hybrid approach that integrates quantum routines in the classical Branch & Price algorithm for large integer linear programs. We test this approach on a smart charging problem that is modeled as *graph coloring problem*. Our computational results affirm the potential of the hybrid approach while revealing the considerable dependence of the performance gain on the particular instance of the problem.

Important components of variational algorithms can be represented as ZX-diagrams. We demonstrate how the rewriting rules of ZX-calculus can be used to derive the analytical formula for the mean energy of a general Ising model in a QAOA₁ state. Furthermore, we contribute to the theoretical exploration of variational algorithms by extending the ZX-calculus with addition and differentiation of ZX-diagrams. Our inductive procedure for the addition is fully diagrammatic. For the differentiation we suggest two approaches. The first approach is inductive, it leverages our procedure for addition to explicitly represent the product rules. The second approach is resumed in two formulas that are derived from the factored form of parameterized diagrams.

Keywords: Combinatorial optimization, quantum computing, energy management, smart charging, ZX-calculus

Résumé

Le domaine du management de l'énergie implique de nombreux problèmes d'optimisation combinatoire connus pour être difficiles. L'émergence des ordinateurs quantiques suggère de nouvelles approches pour ces problèmes. Pour les machines du futur proche, les heuristiques quantiques variationnelles telles que QAOA, qui peuvent tirer parti de la puissance de calcul des ordinateurs quantiques imparfaits, sont particulièrement prometteuses.

Nous explorons le potentiel des algorithmes quantiques variationnels pour des problèmes d'optimisation issus du domaine de "*smart charging*" des véhicules électriques. Nous considérons deux problèmes inspirés de cas d'utilisation réels. Dans le premier problème, modélisé par *Max-K-Cut*, nous cherchons à planifier un ensemble de charges avec priorités sur plusieurs stations tout en minimisant le temps de complétion pondéré. Dans le deuxième problème, modélisé par *Maximum Independent Set*, nous cherchons à maximiser le nombre de demandes de charge satisfaites sur une seule station tout en respectant les conflits entre les demandes. Pour les deux problèmes, nous développons un protocole expérimental spécifiant *l'encodage* et la *routine d'optimisation des paramètres*. Nos expériences numériques confirment l'intérêt des heuristiques quantiques pour ces problèmes ainsi que la qualité de notre protocole expérimental.

Afin d'étendre l'applicabilité des heuristiques quantiques nous introduisons une nouvelle approche hybride qui intègre des routines quantiques dans l'algorithme classique de Branch & Price pour les programmes linéaires en nombres entiers de grande taille. Nous testons cette approche sur un problème de smart charging qui se modélise comme un *problème de coloration de graphe*. Nos résultats numériques affirment le potentiel de l'approche hybride tout en révélant la variabilité du gain de performance en fonction de l'instance particulière du problème.

Les parties importants des algorithmes variationnels peuvent être représentées sous forme de diagrammes ZX. Nous démontrons comment les règles de réécriture du ZX-calculus peuvent être utilisées pour dériver la formule analytique de l'énergie moyenne d'un modèle d'Ising dans un état QAOA de profondeur 1. De plus, nous contribuons à l'exploration théorique des algorithmes variationnels en étendant le ZX-calcul avec addition et différenciation de ZX-diagrammes. Notre procédure inductive pour l'addition est entièrement graphique. Pour la différenciation, nous suggérons deux approches. La première approche est inductive, elle s'appuie sur notre procédure d'addition pour représenter explicitement les règles du produit. La deuxième approche est résumée dans deux formules qui sont dérivées de la forme factorisée des diagrammes paramétrés.

Mots-clés: Optimisation combinatoire, algorithmes quantiques, management de l'énergie, smart charging, ZX-calculus

A.3 Résumé étendu (en français)

Le domaine de la gestion de l'énergie traite de nombreux problèmes d'optimisation qui sont difficiles. Trouver de meilleures solutions à ces problèmes peut réduire considérablement les coûts, améliorer la qualité du service ou résoudre des problèmes opérationnels émergents tels que l'intégration de sources d'énergie renouvelables. Cette thèse est dédiée à l'application d'algorithmes quantiques à des problèmes d'optimisation combinatoire issus du domaine de la "recharge intelligente" - un domaine qui couvre les problèmes liés à la recharge des véhicules électriques.

De nombreux problèmes de charge intelligents sont modélisés comme des problèmes d'optimisation combinatoire NP-difficiles. Cela implique que (dans le pire des cas) la recherche de la solution optimale prend un temps exponentiel. Par conséquent, en pratique de tels problèmes sont résolus de façon approchée. Une solution approchée peut être calculée par deux types d'algorithmes : ceux avec une qualité d'approximation théoriquement démontrée (appelés algorithmes d'approximation) et ceux qui performant bien en pratique (appelés heuristiques).

Pour ces problèmes il est probable que l'utilisation des *algorithmes quantiques* donne un avantage soit en termes du temps d'exécution, soit en termes de la qualité de la solution obtenue.

L'idée d'utiliser des effets quantiques pour améliorer les calculs des propriétés des systèmes physiques (et les calculs en général) a été suggérée indépendamment par Richard Feynman [Feynman, 1986], Yuri Manin [Manin, 1980] et Paul Benioff [Benioff, 1982]. Des machines qui exploitent les principes de la mécanique quantique comme *l'intrication* et *la superposition* sont manifestement plus puissantes que les machines de Turing classiques. Pour le moment, les deux résultats les plus impressionnants sont *l'algorithme de Grover* pour une recherche dans une base de données non structurée [Grover, 1996] et l'algorithme de factorisation des entiers découvert par Shor [Shor, 1995].

D'un point de vue très général la programmation d'un ordinateur quantique consiste à combiner *petites briques* pour préparer une distribution de probabilité qui est pertinente pour le problème en question. Il n'est pas évident de concevoir des algorithmes dans un tel framework. En partie à cause de cela, le domaine de l'informatique quantique se concentre beaucoup sur les algorithmes variationnels qui ne proposent pas des programmes fixes mais plutôt des modèles entraînaables qui sont optimisés pour trouver de bonnes solutions [Preskill, 2018].

Les algorithmes variationnels sont bien adaptés au ainsi appelé NISQ-hardware (Noisy Intermediate Scale Quantum) qui propose entre 10^2 et 10^3 qubits. En raison de la taille limitée et du niveau de bruit élevé, nous ne nous attendons pas à ce que les ordinateurs NISQ exécutent les célèbres algorithmes de Grover et Shor sur des problèmes de tailles pratiquement intéressantes [Babbush *et al.*, 2021]. En plus, il est probable que l'accélération polynomiale des algorithmes [Montanaro, 2020, Ambainis, 2005, Cerf *et al.*, 2000, van Apeldoorn *et al.*, 2020] sera anéanti par les schémas de correction du bruit [Babbush *et al.*, 2021].

Nous suggérons que le meilleur chemin pour avoir un avantage avec les machines NISQ est de combiner les algorithmes variationnels avec les routines classiques dans les *procédures hybrides*.

L'AA initialise le système dans un état fondamental facile à préparer de certains hamiltoniens H initialiser, qui évolue sous l'hamiltonien dépendant du temps qui passe progressivement de H initialiser au H final où l'état fondamental de H final correspond à la solution du problème d'optimisation.

Algorithmes quantiques pour l'optimisation combinatoire: Dans ce travail, nous considérons le soi-disant algorithme adiabatique (qui renvoie une solution exacte pour un problème

d'optimisation) et ses proches cousins - le recuit quantique et QAOA donné. L'algorithme adiabatique The initialise le système dans un état fondamental d'un certains Hamiltonien H_{init} facile à préparer (appelés Hamiltonian de mélange), laisse évoluer le système sous l'Hamiltonien qui passe progressivement de H_{init} à H_{final} où l'état fondamental de H_{final} correspond à la solution du problème d'optimisation.

Le théorème adiabatique postule que si l'évolution est suffisamment lente (par rapport à la différence entre l'énergie de l'état fondamental et l'énergie du premier état excité habituellement appelé *écart spectral*), le système reste proche de l'état fondamental instantané pendant tout le processus [Jansen *et al.*, 2007, Amin, 2009a]. Par conséquent, avec une forte probabilité, le résultat de mesure de l'état final sera l'état fondamental de H_{final} - précisément la solution exacte du problème initial.

Comme la condition d'être "suffisamment lent" est généralement difficile à satisfaire en pratique (ou même à évaluer), l'Algorithme Adiabatique reste essentiellement une construction théorique. Il a cependant inspiré plusieurs heuristiques polynomial telles que le recuit quantique et le QAOA [Farhi *et al.*, 2014a]

Le *recuit quantique* modifie l'hamiltonien de la même manière que l'algorithme adiabatique mais beaucoup plus rapidement que ne le permet la condition adiabatique. L'évolution est effectuée sur les machines spéciales appelées quantum annealers, et le temps d'exécution est généralement limité par les caractéristiques de la machine. Le recuit quantique a été appliqué à de nombreux problèmes d'optimisation tels que le voyageur de commerce et la coloration de graphes[Rieffel *et al.*, 2014], reconnaissance d'images [Venturelli *et al.*, 2015] et la recherche de l'état fondamental du modèle de Sherington-Kirkpatrick [Venturellet coll., 2015]. Essentiellement, l'idée derrière le recuit quantique consiste à utiliser les fluctuations quantiques afin d'explorer l'espace des solutions de façon efficace.

Contrairement au recuit quantique qui nécessite une machine analogique spécifique, le QAOA adapte les idées de l'algorithme adiabatique aux ordinateurs quantiques universels basés sur les circuits [Farhi *et al.*, 2014a]. La dynamique QAOA n'est pas régie par une interpolation continue entre deux hamiltoniens, mais par une séquence de pulsions ("bangs") qui saute entre $H_{initial}$ et H_{final} . Le nombre total des pulsions $p \in \mathbb{N}$, appelé *profondeur*, est prédéfini par l'utilisateur tandis que les durées de bangs sont optimisées directement à l'intérieur de l'algorithme par une routine classique. C'est pour cela que l'algorithme est dit variationnel. Il est connu que pour le grand nombre de couches $p \rightarrow \inf$ l'algorithme atteint l'optimum exact. De plus, contrairement au recuit quantique [Crosson *et al.*, 2014], la croissance de la qualité de la solution en temps est monotone.

Récemment, il y a eu un progrès remarquable dans la mise en œuvre expérimental des quantum annealers : 5000 les qubits sont accessibles sur *DWave Advantage system 6.1* [Systems, 2022] par rapport à des 127 qubits sur l'ordinateur quantique universelle IBM Eagle [Dial, 2022]. Par conséquent, le recuit quantique se prête mieux à l'analyse expérimentale alors que en raison du manque d'outils, l'analyse théorique de l'algorithme est nettement plus difficile [Braida *et al.*, 2022, Moosavian *et al.*, 2022].

D'autre part, pour QAOA de nombreux résultats théoriques ont été obtenus. Entre ces résultats, les plus importants sont la suprématie [Farhi and Harrow, 2016] et l'universalité [Lloyd, 2018]. Certaines garanties de performances ont été démontrées pour des instances spéciales de MaxCut [Farhi *et al.*, 2014a, Hadfield, 2018, Wurtz and Love, 2021, Marwaha, 2021, Basso *et al.*, 2022] et E3LIN2 [Farhi *et al.*, 2014b] ainson que les résultats limitants [Bravyi *et al.*, 2020, Farhi *et al.*, 2020, Chou *et al.*, 2021], bien qu'aucun avantage quantique n'ait été prouvé.

Le dernier algorithme considéré est le QAOA récursif (RQAOA). Le RQAOA a été initialement conçu pour traiter le problème de *localité* qui a souvent été mentionné comme la condi-

tion limitante pour la performance QAOA [Bravyi *et al.*, 2020, Farhi *et al.*, 2020, Hastings, 2019, Chou *et al.*, 2021]. RQAOA procède par une élimination récursive de variables, où à chaque étape la variable à éliminer est sélectionnée en utilisant l'état optimal de QAOA pour le problème restreint. La promesse derrière est que, contrairement au QAOA à profondeur constante, le RQAOA finit par voir la structure globale du problème. Du point de vue expérimental, un fait particulièrement intéressant à propos du RQAOA est que pour $p = 1$ il peut être efficacement simulé sur un ordinateur classique [Bravyi *et al.*, 2020, Egger *et al.*, 2021].

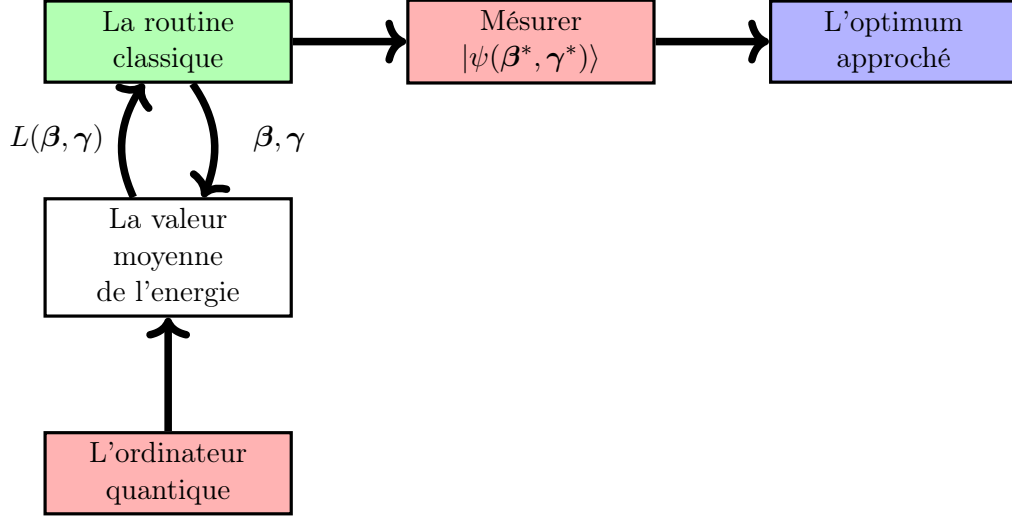


Figure A.2: Schema d'interaction entre la partie classique et la partie quantique dans QAOA

La formulation QUBO: Afin d'appliquer des algorithmes quantiques à des problèmes d'optimisation combinatoire, nous devons d'abord associer l'espace des solutions S à l'ensemble $|x\rangle$, $x \in \{0, 1\}^n$ des états de base d'un système à n qubits. Ensuite, nous devons définir un Hamiltonien diagonal H , telle que pour chaque solution réalisable $x \in \{0, 1\}^n$, la valeur propre de H corresponde à la valeur de la fonction à optimiser:

$$H|x\rangle = f(x)|x\rangle, \quad |x\rangle \in S_b \quad (\text{A.26})$$

Ce mapping est évident pour les problèmes quadratiques sur les variables binaires et sans contraintes (appelés QUBO) et relativement facile pour beaucoup de problèmes d'optimisation NP-difficiles [Lucas, 2014, Alidaee *et al.*, 1994, Kochenberger *et al.*, 2013, Alidaee *et al.*, 2008].

Lorsqu'une formulation du problème d'optimisation dépasse le cadre QUBO (elle a les contraintes ou les variables non-binaires), un traitement spécial est nécessaire pour capturer dans un modèle QUBO ces éléments. En informatique quantique, il existe deux approches pour intégrer les contraintes. Le premier est la pénalisation, soit la modification de la fonction objectif d'une manière qui rend les solutions irréalisables sous-optimales. Pour une contrainte $x \in S$ la pénalité est une fonction $g(x) : \{0, 1\}^n \rightarrow \mathbb{R}^+$ tel que: $g(x) = \begin{cases} 0, & x \in S \\ \geq 1, & x \notin S \end{cases}$.

La deuxième technique pour gérer les contraintes consiste à remplacer l'Hamiltonien de mélange pour maintenir l'évolution dans le sous-espace réalisable [Hen and Spedalieri, 2016, Hen and Sarandy, 2016, Hadfield *et al.*, 2019].

Le recharge intelligente avec les algorithmes quantiques Nous avons sélectionné deux cas d’usage réels pour évaluer les approches quantiques. La première concerne l’ordonancement: le but est d’attribuer un ensemble de charges à différentes bornes de recharge. Dans le deuxième problème, il faut sélectionner un sous-ensemble de demandes de charge à satisfaire tout en respectant les conflit entre les demandes.

Dans le *problème d’ordonnement*, on a un ensemble de demandes (non-préemptif) de charge J , $|J| = n$. Chaque charge $j \in J$ a un temps d’exécution fixe $t_j \in \mathbb{R}^+$ et une *priorité* $w_j \in \mathbb{R}^+$. L’objectif est de distribuer les charges entre un ensemble M de m bornes de recharge identiques. Dans le cas trivial avec une borne un algorithme polynomial découle directement de la règle de Smith [Smith, 1956]. Cependant, pour des instances avec les priorités positives générales le problème est NP-difficile au sens ordinaire quand $m \geq 2$ [Skutella and Woeginger, 2000] et NP-difficile au sens fort quand m fait partie de l’entrée du problème [Garey and Johnson, 1990]. Néanmoins, du point de vue de l’approximation, le problème est simple - pour un nombre fixe de stations m le problème est pseudopolynomial et admet un FPTAS [Sahni, 1976, Woeginger, 2001].

Le problème peut être modélisé comme la recherche de Maximum- m -Cut dans un graph pondéré [Skutella, 1998], et pour le cas ou $m = 2$ cette formulation est facilement transformable dans la forme QUBO [Alidaee *et al.*, 1994]. Le problème Maximum-2-Cut est NP-difficile, en plus il est NP-difficile à approximer la solution avec un facteur supérieur à 0.941... [Arora *et al.*, 1998]. Le meilleur algorithme d’approximation pour Maximum-2-Cut a le ratio $\alpha_{GW} \geq 0.87856$ [Goemans and Williamson, 1995]. De plus, sous la *Unique Game Conjecture* [Khot, 2002] le ratio 0.87856 est optimal [Khot *et al.*, 2007].

Une formulation naturelle du problème Max- m -Cut utilise des variables entières $x_i \in [1, \dots, m]$. correspondants aux couleurs attribuées aux nœuds:

$$\max_{x \in [1, \dots, m]^n} \sum_{(u,v) \in E} e_{u,v} I(x_u \neq x_v) \quad (\text{A.27})$$

Pour mapper la formulation A.27 à QUBO, nous avons suggéré d’utiliser un *encodage binaire*. Pour $m = 2^l$ chaque variable entière $x_i \in [0, \dots, m-1]$ est mappé à une séquence de $l = \log_2 m$ variables binaires. Une chaîne de bits $\mathbf{b}_i = [b_0, \dots, b_{l-1}]$ correspond à l’entier $x_i = \sum_{0 \leq j \leq l-1} 2^j b_j$.

Nous avons analysé numériquement les performances de QAOA sur des graphes correspondant au problème de recharge intelligente. Les instances ont été générées à partir d’un ensemble de données réels contenant environ 2250 charges réalisées au cours du mois de mai 2017 sur des points de charge du réseau Belib’s [Bel, 2017]. Nous avons utilisé le phénomène de concentration pour prédire de bons points d’initialisation pour différentes taille des instances n . Pour l’optimisation des paramètres, nous avons comparé différentes méthodes d’optimisation locales ; il s’est avéré que le *Nelder-Mead* est le meilleur choix à la fois en termes de qualité de la solution et de temps d’exécution. Comme prévu, nous avons observé que la performance QAOA augmente avec la profondeur p . De manière surprenante, nous avons également remarqué que le rapport d’approximation s’améliore avec la taille n de l’instance.

Dans le problème de sélection des demandes de charge à satisfaire, chaque demande (non préemptive) doit être exécuté dans un interval prédéfini $[T_j^0, T_j^0 + t_j]$. Si accepté, un demande j apporte un profit $p_j \in \mathbb{R}^+$. Le décideur vise à sélectionner une *planification valide* avec le profit le plus élevé. Une planification est valide si les tâches sélectionnées sont simultanément satisfaisables, i.e. leurs interval ne se chevauchent pas $[T_i^0, T_i^0 + t_i] \cap [T_j^0, T_j^0 + t_j] \neq \emptyset$. Nous considérons ici ce problème avec des contraintes de groupe en plus qui indiquent que de demande de même groupe de ne peuvent pas être satisfait ensemble.

Ce problème équivaut à la recherche d’un stable maximal (MIS) dans un graphe. Le MIS est difficile à approximer : selon [Håstad, 1999, Zuckerman, 2007] pour toute ϵ il n’y a pas

d'algorithme polynomial retournant pour chaque instance un stable S^* tel que $|S_{max}|/|S^*| \leq n^{1-\epsilon}$ sauf si $P = NP$.

La formulation QUBO pour le problème de stable maximal est:

$$\max \sum_{v \in V} x_v - \lambda \sum_{(u,v) \in E} x_u x_v \quad (\text{A.28})$$

$$x \in \{0, 1\}^{|V|} \quad (\text{A.29})$$

where $\lambda = 2$.

Afin d'évaluer les performances de l'heuristique quantique sur le problème MIS, nous avons créé un benchmark qui contient des graphes réguliers aléatoires avec des degrés dans $[3, 5, 7, 9]$ et des tailles allant de 100 à 1000 noeuds. Nous avons également inclus des graphes du benchmark DIMACS et des instances réelles issues d'un simulateur de scénarios de demande de recharge développés en interne par EDF. Comme RQAOA en profondeur 1 peut être simulé efficacement, nous l'avons utilisé pour évaluer les performances sur de grandes instances.

Nous avons comparé le RQAOA à l'algorithme d'approximation [Boppana, 1990] sur 121 graphes réguliers aléatoires avec jusqu'à 1000 sommets. Nous avons observé que sur tous sauf 7 instances RQAOA renvoie une solution au moins aussi bonne que celle trouvée par la routine classique. Sur le benchmark DIMACS, bien que nous n'ayons pas été en mesure de surpasser définitivement les heuristiques classiques, les solutions obtenues étaient encore compétitives dans de nombreux cas. De plus, nous pensons que les performances de l'algorithme peuvent être considérablement augmentées si l'on considère la profondeur plus élevée $p > 1$. En revanche, nous avons observé que la qualité de la solution n'est pas robuste d'une instance à l'autre. Par conséquent, la sélection des bons cas d'utilisation qui conduisent à des instances bien adaptées à l'heuristique quantique reste un défi extrêmement important pour les algorithmes quantique du futur proche.

Approches hybrides: Selon les estimations actuelles, le matériel quantique du futur proche aura un nombre relativement faible de qubits avec une connectivité limitée [Preskill, 2018]. Par conséquent, pour la plupart des applications matérielles quantiques, il est nécessaire de combiner des routines classiques et quantiques dans des approches hybrides. Les routines classiques peuvent être utilisées en prétraitement [Choi, 2008], en posttraitement ainsi qu'être directement intégrées dans de tels algorithmes variationnels comme QAOA et VQE [Braine *et al.*, 2021] ou les utiliser comme sous-programme [Yonaga *et al.*, 2020, Ohzeki, 2020].

Outre la gestion des contraintes matérielles, l'hybridation permet de résoudre les formulations plus compliquées qu'un simple QUBO. Dans cette thèse notre objectif est de concevoir un approche hybride quantique-classique pour les programmes linéaires en nombre entiers sur un grand nombre des variables. Pour cela, nous combinons des routines quantiques avec la technique de Branche Price qui s'attaque à un grand nombre de variables avec la génération de colonnes [Vanderbeck, 1994]. Nous testons notre approche sur la formulation étendue du problème de coloration de graphe (problème maître) [Mehrotra and Trick, 1996]:

$$(\text{GC-s}) : \quad \min \sum_{s \in I(G)} \lambda_s \quad (\text{A.30})$$

$$\sum_{\substack{s \in I(G) \\ v \in s}} \lambda_s \geq 1, \quad \forall v \in V \quad (\text{A.31})$$

$$\lambda_s \in \{0, 1\}, \quad \forall s \in I(G) \quad (\text{A.32})$$

où $I(G)$ est l'ensemble de tous les stables dans le graphe. Suivant l'approche suggérée dans [Mehrotra and Trick, 1996], nous ne considérons que le sous-ensemble restreint $I' \subsetneq I(G)$ de variables et ensuite ajoutons des variables «prometteuses» avec la routine de *pricing*.

Dans [Svensson *et al.*, 2021] QAOA est utilisé comme solveur heuristique pour le problème maître restreint. Au lieu de cela, nous suggérons une approche différente qui intègre l'heuristique quantique dans l'étape de pricing de la génération des colonnes. Pour le problème de coloration, le sous-problème de pricing équivaut à la recherche d'un stable maximal pondéré (MWIS). Dans notre approche hybride, l'instance de MWIS est abordée avec une procédure en trois volets qui combine l'heuristique classique de [Held *et al.*, 2012], RQAOA et l'algorithme exact.

Une expérience numérique sur des instances aléatoires de densités différentes montre qu'il est effectivement profitable dans certaines cas d'utiliser la méthode hybride au lieu de l'approche classique traditionnelle. Malheureusement, pour les instances issues du problème de sélection des tâches de charge, nous avons observé qu'il n'y a pas besoin de RQAOA car l'heuristique classique trouve toujours la solution du problème de pricing quand une telle solution existe.

ZX-calculus: ZX-calcul, initialement introduit dans [Coecke and Duncan, 2011], est un *langage graphique* qui permet de raisonner sur l'informatique quantique. Dans ce langage, les calculs complexes sur les qubits sont représentés par *diagrammes* fabriqués à partir de générateurs élémentaires. Chaque diagramme correspond à une transformation linéaire entre des espaces de Hilbert d'états des qubits. Un ensemble compact de règles de réécriture permet de transformer des diagrammes en diagrammes équivalents. L'avantage notable du calcul ZX par rapport à d'autres représentations est que dans ce langage, les calculs peuvent être effectués *entièrement avec les transformations graphiques* [Hadzihanovic *et al.*, 2018, Jeandel *et al.*, 2018b].

Grâce à sa flexibilité, ZX-calculus est largement utilisé pour adresser différents problèmes d'informatique quantique. Par exemple, le calcul ZX a permis la dérivation de résultats importants dans le domaine de *Measurement-based Quantum Computing* (MBQC) [Duncan and Perdrix, 2010, Kissinger and van de Wetering, 2019] ainsi que pour l'optimisation des circuits [Cowtan *et al.*, 2020, de Beaudrap *et al.*, 2020, Kissinger and van de Wetering, 2020b, Duncan *et al.*, 2020] et pour la conception des codes correcteurs d'erreurs [Garvie et Duncan, 2018], ainsi que l'analyse et la compilation de codes de surface [Garvie and Duncan, 2018, Horsman, 2011, de Beaudrap and Horsman, 2020, Hanks *et al.*, 2020].

Cependant, les applications du calcul ZX aux algorithmes variationnels sont jusqu'à présent limitées. Nous pensons que la raison pour laquelle ils sont encore inexplorés avec les moyens du calcul ZX est l'absence d'un moyen pratique de différencier les diagrammes paramétrés. En effet, les blocs de construction de base des algorithmes variationnels sont des circuits paramétrés, et la recherche de valeurs de paramètres optimales est une partie cruciale de ces algorithmes. En théorie, trouver des paramètres optimaux peut être NP-difficile [Bittel and Kliesch, 2021]. En pratique, la recherche est généralement effectuée par des méthodes d'optimisation numériques classiques et la plupart d'entre elles utilisent des dérivées [Guerreschi and Smelyanskiy, 2017].

La principale difficulté de différenciation des ZX-diagrammes provient du règle du produit: $\partial fg = \partial f g + f \partial g$ qui consiste à ajouter deux termes. Cette règle est cruciale pour évaluer la dérivée de la composition séquentielle et du produit tensoriel. Comme les diagrammes ZX sont définis de manière inductive avec ces deux compositions, la dérivée d'un diagramme complexe pourrait être calculée à partir des dérivées de ses parties si nous étions autorisés à utiliser la règle du produit, et donc les diagrammes de somme.

Les œuvres [Zhao and Gao, 2021, Toumi *et al.*, 2021] utilisent des sommes explicites de diagrammes pour représenter la dérivée des diagrammes avec plusieurs occurrences du paramètre.

L'inconvénient majeur de cette approche est qu'il n'y a pas de règles pour manipuler des sommes formelles des ZX-diagrammes. Par conséquent, nous ne pouvons pas exploiter pleinement la puissance du calcul graphique. Dans cette thèse, nous proposons une approche où la dérivée d'un paramètre ZX-le schéma est un autre ZX-diagramme. On évite ainsi l'extension de la signature avec des sommes formelles. Afin d'aborder les sommes qui apparaissent dans la règle du produit, nous introduisons une technique originale pour effectuer l'addition des diagrammes entièrement dans la ZX-calcul. Pour cela, on utilise des diagrammes spéciaux appelés états contrôlés [Jeandel *et al.*, 2019]:

Definition A.3.1 (Etats contrôlé [Jeandel *et al.*, 2019]) Une diagramme ZX avec une entrée et n sorties est un état contrôlé si $\llbracket C \rrbracket |0\rangle = \sum_{x \in \{0,1\}^n} |x\rangle = \left[\begin{array}{c} \text{diagram with } n \text{ green circles} \\ \vdots \\ n \end{array} \right]$.

Nous proposons une voie inductive (nommée controlizer) pour représenter chaque ZX-diagramme par un tel état.

Definition A.3.2 (Controlizer) On dit qu'un mapping C qui associe à chaque diagramme avec n entrées et m sorties une autre diagramme avec une entrée et $n + m$ sorties est un controlizer si pour toute diagramme ZX-diagram D :

- $C(D)$ est un état contrôlé
- L'état correspond à la diagramme D dans le sens que:

$$\llbracket D \rrbracket = \left[\begin{array}{c} \text{diagram with } n+m \text{ inputs and } n \text{ outputs} \\ \vdots \\ n \end{array} \right] \quad (A.33)$$

Comme on sait sommer des états contrôlés [Jeandel *et al.*, 2019] l'addition pour les diagrammes arbitraires suit. Malheureusement, notre démarche inductive conduit à de grands diagrammes, même lorsque les diagrammes initiaux sont assez simples. Ce défaut est cependant conforme à l'intuition. En effet, contrairement aux compositions séquentielles et parallèles, l'addition n'est pas une opération physique, il n'est donc pas surprenant qu'elle ne puisse pas être facilement intégrée dans le cadre des schémas ZX. Du côté positif, on peut s'appuyer sur la puissante théorie des équations du ZX-calcul pour simplifier, quand c'est possible, les diagrammes représentant la somme de diagrammes.

Une approche alternative pour l'addition dans le calcul ZXW algébrique a été présentée dans [Yeung *et al.*,]. Il s'appuie également sur les formes contrôlées du diagramme, mais leurs formes contrôlées sont différentes des nôtres.

Une définition inductive de la dérivée est obtenue par une représentation explicite des règles du produit.

Dans un but de donner une technique prête à l'emploi pour la différenciation, nous proposons un moyen simple et pratique de calculer la dérivée pour la famille de diagrammes linéaires $ZX(\beta)$ [Jeandel *et al.*, 2019], c'est-à-dire des diagrammes où les angles ne peuvent dépendre que linéairement (avec des coefficients entiers) d'un paramètre β . La plupart des circuits utilisés pour

les algorithmes variationnels appartiennent $ZX(\beta)$ et nous pensons que nos formules rendront leur analyse beaucoup plus simple.

Tout d'abord, nous observons que chaque diagramme dans $ZX(\beta)$ peut être transformé sous la forme

$$D(\beta) = \begin{array}{c} \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array} \quad \begin{array}{|c|} \hline X_\beta(n, m) \\ \hline \dots \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline D_2 \\ \hline \dots \\ \hline \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline \dots \\ \hline D_1 \\ \hline \dots \\ \hline \end{array} \quad \begin{array}{|c|} \hline \beta \quad \dots \quad \beta \quad -\beta \quad \dots \quad -\beta \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline D_2 \\ \hline \dots \\ \hline \end{array} \end{array} \quad (A.34)$$

où n, m sont les nombres entiers et D_1, D_2 ne contiennent pas β .

Définition A.3.3 *Etant donné une diagramme $D(\beta) = D_2 \circ (D_1 \otimes X_\beta(n, m))$ dans la forme (A.34), on a*

$$\partial_{ZX}[X_\beta(n, m)] = \partial_{ZX} \left[\begin{array}{c} \beta \quad \dots \quad \beta \quad -\beta \quad \dots \quad -\beta \\ \underbrace{\hspace{1.5cm}}_n \quad \underbrace{\hspace{1.5cm}}_m \end{array} \right] \quad (A.35)$$

$$= \begin{array}{c} \begin{array}{|c|} \hline \pi \quad \pi \quad \pi \\ \hline \end{array} \quad \begin{array}{|c|} \hline \otimes 3 \\ \hline \end{array} \quad \begin{array}{|c|} \hline \otimes n+m \\ \hline \end{array} \quad \begin{array}{|c|} \hline \pi \quad \pi \quad \pi \quad \dots \quad \pi \quad \pi \quad \pi \quad \dots \quad \pi \quad \pi \quad \pi \\ \hline \end{array} \quad \begin{array}{|c|} \hline \beta \quad \dots \quad \beta \quad -\beta \quad \dots \quad -\beta \\ \hline \end{array} \end{array} \quad (A.36)$$

Une définition d'une dérivée similaire à nos formules a été obtenue dans le travail indépendant [Wang and Yeung, 2022]. Ce travail utilise W-spiders pour gérer les règles du produit. Contrairement à notre résultat, la différenciation diagrammatique présentée dans [Wang and Yeung, 2022] mappe un diagramme ZX à un diagramme d'un autre langage appelé ZX-calculus algébrique [Wang, 2020]. Le calcul ZX algébrique est pratique pour représenter des nombres complexes arbitraires, donc leur procédure de différenciation peut gérer des familles de diagrammes plus générales que $ZX(\beta)$. Cet avantage s'accompagne du coût de l'abandon de l'héritage du calcul ZX vanille qui est de loin le calcul graphique le plus populaire pour le calcul quantique.