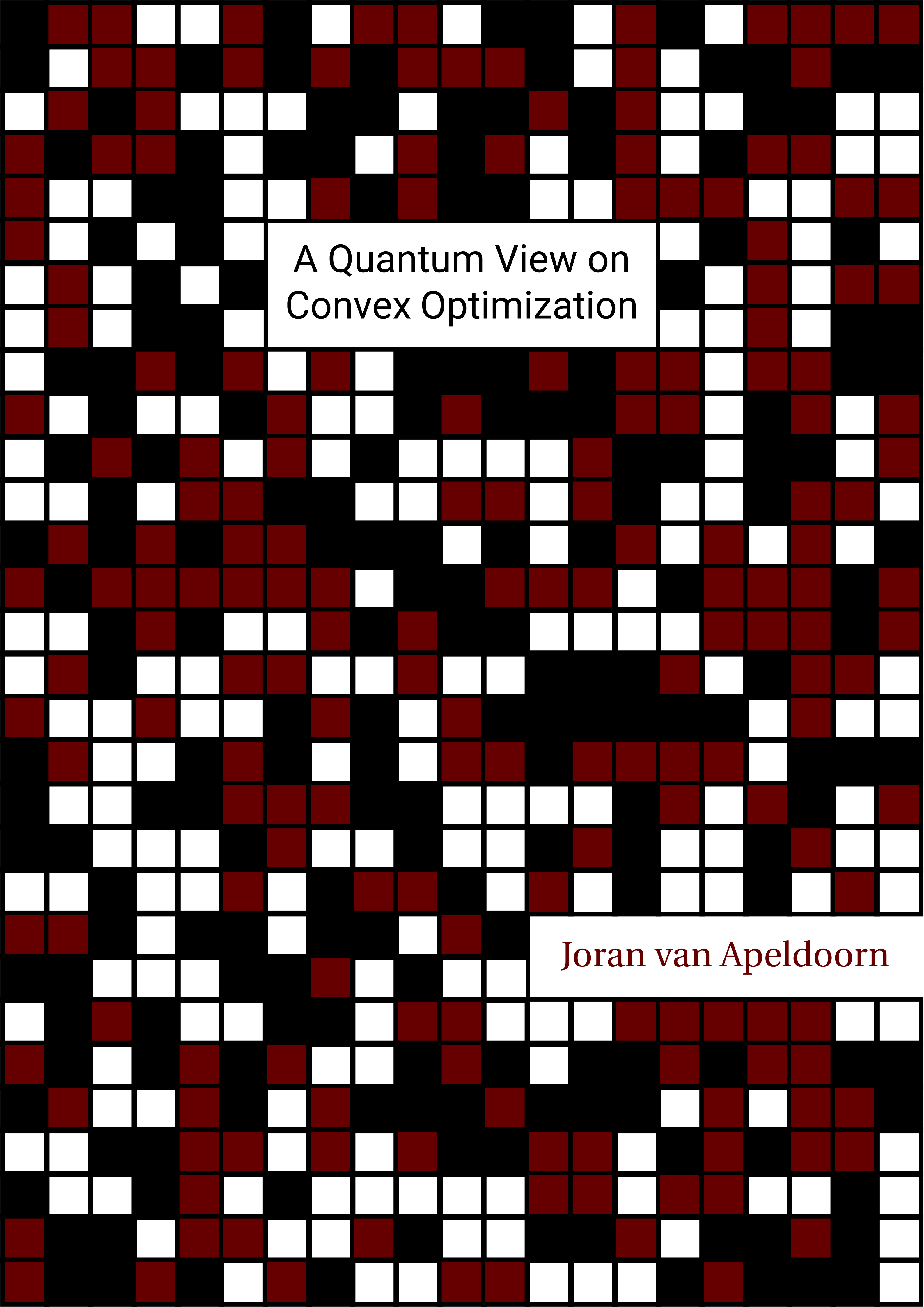
A quantum view on convex optimization

van Apeldoorn, J.T.S.



A Quantum View on Convex Optimization

Joran van Apeldoorn

A Quantum View on Convex Optimization

Joran van Apeldoorn

ILLC Dissertation Series DS-2020-04

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

CWI
Centrum Wiskunde & Informatica

QuSoft
Research Center for Quantum Software

The investigations were supported by the Netherlands Organization for Scientific Research (NWO), TOP Grant number 617.001.351, and by QuantERA project QuantAlgo 680.91.034.

For the quote by Rosa Luxemburg used as an epigraph, see [Lux22, p. 109].

The cover pages of the printed copies where individually generated using a quantum random number generator.

Printed and bound by the Paper Jam Collective.
ISBN: 9789082347289

A Quantum View on Convex Optimization

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op donderdag 6 februari 2020, te 14.00 uur

door

Joran Tomas Siyé van Apeldoorn

geboren te Amsterdam

Promotiecommisie

Promotores: Prof. dr. R.M. de Wolf
Universiteit van Amsterdam

Prof. dr. M. Laurent
Tilburg University

Overige leden: Prof. dr. N. Bansal
Centrum Wiskunde & Informatica

Prof. dr. H.M. Buhrman
Universiteit van Amsterdam

Prof. dr. E.M. Opdam
Universiteit van Amsterdam

Prof. dr. B.M. Terhal
TU Delft

Dr. M. Ozols
Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

“Freiheit ist immer Freiheit des anders Denkenden”
— Rosa Luxemburg

Table of Contents

Acknowledgements

I want to start by thanking my advisors, Ronald and Monique. Ronald's help with my understanding of quantum computing started some time before we knew each other, when I read his excellent lecture notes during my bachelor project. During my time at CWI Ronald's door has always been open, he often pointed out interesting problems for us to work on, and he was a great help while navigating a complex and new research area. We often clashed due to differences in style in mathematical discussions, and although frustrating in the moment, in hindsight I am glad he kept pushing me to be clearer and more precise, and my improvements in this area are probably the most important thing I will take away from the past four years (I will never dare to write a wrongly-hyphenated sentence again). Although Monique and I have worked together less over the last years, her influence on this dissertation is still very present. My first steps in the world of optimization were through her course on semidefinite optimization, and I am very glad that I got introduced to this wonderful area of research. I especially want to thank Monique for the insightful comments on earlier versions of this dissertation, they often helped me see a different side of my own writing and helped me clear up the many unclear statements.

I also want to thank the members of my thesis committee, Nikhil Bansal, Harry Buhrman, Māris Ozols, Eric Opdam, and Barbera Terhal, for taking the time to read my dissertation, and for sending me useful comments. I look forward to discussing my dissertation with you on the 6th of February.

Next, I would like to thank Leen Torenvliet for shaping my academic career. My first introduction to algorithms and complexity came from Leen's course on the topic. When I asked him for a bachelors project he introduced me to Harry, and in doing so, led me to the field of quantum computing. Over the years I have had the pleasure of working with Leen many times, and of having many much needed coffee breaks together. Hopefully we can continue this over the coming years.

I want to express my sincere gratitude towards my co-authors András and Sander, both of whom also became good friends of mine over the last couple of years. András and I shared many offices and hotel rooms over the years, and had many interesting discussions, both about mathematics and many other topics. We complemented each other in a very productive way and I hope we get the chance to work together again. Sander is a great stabilizing force in research meetings, and taught me much about optimization. I really enjoyed working together on organizing the QSC junior day over the past few weeks as well.

My time at CWI would not have been the same without all my great colleagues: Alex, Arjan, Bas, Álvaro, Casper, Chris, Chris, Chris, Farrokh, Florian, Freek, Harold, Ido, Isabella, Jan, Jana, Jeroen, Jonas, Jop, Joris, Koen, Michael, Peter, Simon, Srini, Stacey, Subha, Teresa, Tom, Yfke, and Yinan. Thanks for many discussions, coffee breaks, cakes, qusoft lunches, junior meetings, and games of foosball.

I would like to thank Ashley Montanaro for welcoming me in Bristol. During my time there I had many interesting discussions, and I am glad I got to meet Noah, Paul, Stephen, Dominic, Chris and many others.

Over the years I have had the pleasure of fighting alongside many great people for the things that we believe in. I will not be able to name them all, partially for security reasons, but I cannot go on without mentioning some of them. Pretty Boy, we have seen way too little of each other recently, I miss the stories about penguins. Maks, I am happy that we recently found each other again. Rudolf, you have inspired me to never stop fighting for the scientific values. To those with whom we had the crazy idea to take on the national government in a referendum, I am surprised we all survived. I especially want to thank Tijn, you have become a valued friend, and, for a few hours a week, a great teacher. Finally I want to thank the board of Alert, over the last few months I have learned a lot, and I am proud to be part of a group that has such a wide reach.

Special thanks to my paranymphs, Lotte and Okke. Not just for helping me out with my defense, but also for all the times you have been there for me over the years. Okke, over the years you have become my best friend. Whether we were brewing beer, drinking it, taking road trips, or coming up with Suske & Wiske titles in a swimming pool, you always managed to make me feel at ease. My recent visit to Glasgow reminded me of how much I miss you in Amsterdam, hopefully you will be able to return here after your PhD. Lotte, somehow it feels like we have known each other forever, even though it only has been a few years. You have quickly become one of my closest friends, and I am glad we got to know each other and that you got me involved with Alert. I look forward to many more days under pink umbrellas, many more glasses of whisky, and much more dancing to Sister Sparrow.

Rosa, thanks for putting up with my absence and stress while writing this dissertation. Your support kept me going, whether trough mental support or just making a cup of tea for me when I crashed down on the couch after working till 11pm. I can't wait to spend many more King's Days hunting for cupcakes.

Finally, I owe a lot to my parents, Nico & Myrna. From a young age you encouraged me to ask questions and join the conversation, you believed that kids are just as capable as adults at understanding the world (if not more). In doing so you sparked a curiosity that led me to science, formed my life in many different ways, and will continue to do so for many more years.

Joran van Apeldoorn
Amsterdam
December, 2019

Chapter 1

Introduction & Overview

1.1 | Introduction

The 20th century has seen many advances in science and technology. Not only has huge progress been made, often this was done via groundbreaking ideas and radical shifts in our thinking, leading to many new disciplines of science. In this thesis we consider the interplay between three of these disciplines: the field of quantum mechanics, the field of computer science and the subfield of convex optimization.

Quantum mechanics The field of *quantum mechanics* finds it roots in 1900 in the works of Planck [Pla00a; Pla00b], in which he proposes the idea that energy might be *quantized* (meaning it would come in discrete packets, opposed to a continuous spectrum) in order to explain the observations of black-body radiation. This idea turned out to be fruitful and allowed Einstein to explain the photoelectric effect in 1905 [Ein05] and Bohr to explain the spectral lines of the hydrogen atom in 1913 [Boh13]. The notion of discrete photons (the particles of light), however, seemed to contradict experiments that found that light behaves like a wave. This led to the works of Schrödinger [Sch26], Heisenberg, Born and Jordan [Hei25; BJ25; BHJ26] in the 1920s, which introduced the notions of the *wave function* and *matrix mechanics*, fleshing out quantum mechanics and explaining this particle-wave duality.

Light indeed comes in discrete packets (as do other particles), but the position of these packets is not precisely defined. Instead there is a certain *probability* that a particle is at a certain place, leading to a wave-like behavior as these probabilities spread through space. Furthermore, these probabilities do not originate from our lack of knowledge about the world, they are inherent to the world: until observed there is *no real* position of the particle. Instead the system is described by a set of *amplitudes*, one for each possible position in space, from which the corresponding probability can be derived. By the end of the 1930s quantum mechanics grew to one of the most important fields of physics.

Computer science It can be argued that *computer science* finds it roots thousands of years ago, starting with the first abacus. Computer science as we know it today, however, started in the 20th century when Church and Turing built on the work of Gödel and formalized the notion of an *algorithm* [Tur36; Chu36]. They formulated the Church-Turing thesis, stating that any calculation that is possible can be performed by a machine following an algorithm (if given sufficient resources). Turing formalized the idea of a machine, now known as a *universal Turing machine*. During the Second World War computing became of vital importance for military purposes, leading to large advances in computing hardware, the most famous example of which is the work of Turing and others at Bletchley Park that allowed them to break the Enigma codes used by the German army.

After the war the theoretical study of computing continued. In 1965 Edmonds gave the first *polynomial-time algorithm* for finding matchings in general graphs [Edm65] and introduced the notions of *efficient* algorithms (those that run in polynomial time in the input size) and *inefficient* algorithms (those that run in superpolynomial time). He also introduced the notion of the class of problems that have such efficient algorithms, and in doing so he started the field of complexity theory. While computer science was evolving as a theoretical field, an even more impressive achievement has been made in the development of hardware. After the creation of the first working transistor in 1947 by Bardeen, Brattain, and Shockley, progress has been rapid. This progress has for a long time been described best by Moore's law [Moo65], which states that the number of components on a chip and the computational power of chips grows exponentially, doubling every 12 to 18 months. However, Moore's law seems to be coming to an end as transistor sizes are reaching the nanometer scale and quantum mechanical effects start influencing computations.

Convex optimization An important category of problems in mathematics is *optimization problems*, an interesting subset of which are the *convex optimization problems*. Convex optimization problems have many interesting properties, one of which is that a local minimum is also a global minimum; due to this there is no way to "get stuck" in a local minimum that is not the actual optimal solution. Convex optimization as a field leads back to the Second World War, during which Dantzig invented *linear programs* and the *simplex algorithm* as a tool to optimize resource allocation for the army [Dan48]. After the war Von Neumann found a relation between linear programming and zero-sum games, and discovered linear programming duality [Neu47]. In 1979 Khachiyan gave the first polynomial-time algorithm for linear programming, the *ellipsoid algorithm* [Kha79]. While linear programs have many applications, interest also grew for more general optimization. Grötschel, Lovász and Schrijver in 1988 showed the polynomial-time equivalence of many general problems in convex optimization [GLS88]. Goemans and Williamson applied the framework of *semidefinite programming*, a generalization of linear programming, to obtain the first non-trivial polynomial-time approximation algorithm for the MAXCUT problem in 1995 [GW95]. In recent years convex optimization has seen many applications in machine learning and operations research, with growing datasets motivating research into evermore efficient algorithms, often increasing the dependence on the error to allow for linear and sublinear dependence on the input size.

Quantum computing For most of the 20th century science diversified, with more specialist fields required for the more complex questions science faced. However, in the last quarter of the 20th century interdisciplinary research made a comeback. One example of this is the usage of computers in many fields of science. In the natural sciences computers help to understand complex systems by *simulating* the dynamics of these systems, allowing researchers to examine them in a controlled manner. However, classical computers seem to perform badly at simulating the dynamics of quantum mechanics, in part due to the enormous number of amplitudes needed to represent even very simple systems. This led Feynman and Manin to coin the notion of a *quantum computer* in the early 1980s [Fey82; Fey85; Man80; Man99]. They proposed that this new type of computer could use quantum mechanical effects to simulate quantum mechanics.

In 1992 Deutsch and Jozsa constructed a computational problem for which quantum computers are *exponentially better* than deterministic classical computers [DJ92]. Their algorithm inspired Simon's 1994 work where he presents a problem that can be solved exponentially faster on a quantum computer than on a randomized classical computer [Sim97]. In the same year Shor built on top of Simons work to prove his famous result that quantum computers can factor large integers in polynomial time [Sho97], implying that most of modern public-key cryptography could be broken using a quantum computer. Two years later Grover showed that searching could be sped up *quadratically* on a quantum computer [Gro96]. Although Grover's algorithm achieves less of a speedup than the other algorithms mentioned, it can be applied to a far larger range of problems, leading to speedups for many graph problems [DHHM06], minimum-finding [DH96], and much more. In recent years much progress has been made on building quantum computers, increasing the number of *qubits* and their stability. Very recently the first quantum computations was performed that can no longer be simulated on our best classical computers in a reasonable amount of time [Aru+19].

Convex problems in quantum information There also exists a natural connection between quantum mechanics and convex optimization. Both quantum states and quantum measurements correspond to positive semidefinite matrices, making semidefinite programming a natural framework to deal with problems in *quantum information theory*. These problems include finding the optimal measurement to distinguish quantum states [Eld03], computing the fidelity between states [Wat18], and computing the optimal value of nonlocal two-player XOR games [Tsi87; CHTW04]. Semidefinite programming has also been used in the groundbreaking proof of **QIP** = **PSPACE** [JJUW11].

The developments in quantum mechanics, computer science, convex optimization, and the interplay between the fields, have led to many advances in science and technology. However, until recently little research has focused on how the interplay between these three fields might inspire quantum algorithms for convex optimization. This thesis asks the following central question

Can quantum computers solve convex optimization problems faster?

1.2 | Overview

The results presented in this work have been split into two parts. In **Part 1** we will discuss what quantum computers *can do* concerning convex optimization. We will give quantum algorithms for a number of settings in convex optimization and hence upper bound the quantum time and query complexity. In **Part 2** we consider what quantum computers *cannot do* concerning convex optimization. We prove quantum query lower bounds for certain convex optimization problems and show the limitations of some of the techniques introduced in Part 1.

In **Chapter 2** we start by introducing some basic notation and terminology. We then give a short introduction to the basics of quantum computation, followed by an overview of useful quantum algorithmic techniques that will be used throughout this thesis. These include amplitude amplification, phase estimation, amplitude estimation, quantum gradient calculation and the block-encoding framework. We finish the chapter with a general introduction to convex optimization, including some motivations for semidefinite programming and linear programming.

In **Chapter 3** we consider convex optimization in a general setting. Here we have no promise on the structure of the convex problem we are solving, and access to the problem is given in a *black-box* manner. We consider five types of black-box *oracles* for convex sets coming from the classical literature [GLS88] and examine the *quantum* query complexity of reductions between these oracles. In recent work Lee et al. [LSV18] gave a classical algorithm that implements a *separation oracle* using $\widetilde{\mathcal{O}}(n)$ queries to a *membership oracle* (here n denotes the dimension of the space the convex set lies in). We build on and simplify their result, and show that there exists a *quantum* algorithm for the same task that uses only $\widetilde{\mathcal{O}}(1)$ queries, an exponential improvement over the best known classical complexity. Due to a known reduction from optimization to separation [LSW15] we also get as a corollary that only $\widetilde{\mathcal{O}}(n)$ quantum queries are enough for *optimization* using a membership oracle, while the best known classical upper bound is quadratic.

In **Chapter 4** we introduce more structure into the convex optimization problems we are solving by considering *semidefinite programs* (SDPs). As first observed by Brandão and Svore [BS17], the intermediate solutions of certain classical SDP-solving frameworks can be stored as quantum states on a logarithmic number of qubits, allowing for a quadratic speedup in terms of the dimension n and the number of constraints m of the SDP. We start by refining their techniques to bring down the original complexity of $\widetilde{\mathcal{O}}\big(\sqrt{nm}s^2\gamma^{32}\big)$ (where s is the row and column sparsity of the input matrices and γ is the inverse of a scale-invariant error parameter) to $\widetilde{\mathcal{O}}\big(\sqrt{nm}s^2\gamma^8\big)$. We then introduce multiple new techniques and ideas to bring down the complexity further and allow for new input models as well. This results in a complexity of $\widetilde{\mathcal{O}}\big((\sqrt{n}+\sqrt{m})\alpha\gamma^5\big)$ in the general *operator input model*, where $\alpha = s$ for the classic sparse input model.

In **Chapter 5** we restrict to the even more specific set of *linear programming* (LP) problems. In particular we consider quantum algorithms for finding the Nash-equilibrium of two-player *zero-sum games*. Via a reduction these algorithms also apply to linear programming. The presented algorithms build on the techniques developed in Chapter 4, but are more efficient than a direct application of the SDP-solvers to LPs. The proven upper bound is $\widetilde{\mathcal{O}}\left(\sqrt{n+m}\gamma^3\right)$ when the LP has a dense constraint matrix, and $\widetilde{\mathcal{O}}\left(\sqrt{s}\gamma^{3.5}\right)$ when the LP has at most s non-zero entries in each row or column of its constraint matrix.[1] This is the first quantum LP-solver that takes the LP sparsity into account.

In **Chapter 6** we give an overview of different techniques that can be used to prove lower bounds on quantum query complexity. We start by introducing a general model of query algorithms. After this the *polynomial method* by Beals et al. [BBCMW01] is explained. We then state three different forms of the *adversary method*. The first is the basic adversary method due to Ambainis [Amb00], the second is a variation on the first method that allows us to consider the more general phase oracles, and the last method is the general adversary method due to Høyer, Lee and Špalek [HLŠ07].

In **Chapter 7** we return to the setting of Chapter 3 and prove query lower bounds on the relations between the different black-box oracles for a convex set. We first prove a *classical* lower bound of $\Omega(n)$ queries for implementing a *separation oracle* using *membership queries*. This shows that our quantum algorithm from Chapter 3 gives an exponential speedup over the best possible classical algorithm. We then ask whether quantum computers can *optimize* faster using queries to a *separation oracle*. In the setting where an interior point of the convex set is known in advance we prove an $\Omega\left(\sqrt{n}\right)$ lower bound on the number of quantum queries needed. When no such point is known, we are able to prove an $\Omega(n)$ lower bound, matching the classical upper bound of $\mathcal{O}(n)$ from [LSW15]. This leaves as an open question whether convex optimization can be sped up on a quantum computer when an interior point is known.

In **Chapter 8** we consider a *linear* version of *Simon's problem*. For a matrix $A \in \mathbb{F}_p^{n\times n}$ this problem asks to determine whether A is full-rank or not by using matrix-vector product queries of the form $x \mapsto Ax$. The study of this problem is motivated by the open question from Chapter 7, since the *real-valued* version of this problem corresponds to a convex optimization problem. Koiran et al. [KNP07] have shown that the original *non-linear* version of Simon's problem takes at least $\Omega(n)$ quantum queries. We modify their proof to work when restricted to linear functions, and prove that the $\Omega(n)$ lower bound still holds. Even though this is a stronger result, the proof becomes simpler in certain parts.

In **Chapter 9** we discuss the limitations of quantum LP and SDP-solvers. We start with the limits of *specific methods*, including those used in Chapter 4 and Chapter 5. We

[1] This s is different from the s in the SDP case. LPs correspond to SDPs with diagonal input matrices, so the row and column sparseness of the corresponding SDP is always 1.

show that for many natural classes of problems the scale-invariant error parameter can scale linearly with the number of variables and the number of constraints. Due to the polynomial dependence of our methods on this parameter, this limits the number of applications. Another limitation comes from the *sparseness* of the final solutions given by the algorithms. These solutions have a number of non-zero entries that grows at most linearly with the number of iterations of the algorithm. By showing that a solution cannot be sparse, we can lower bound the number of iterations required. As a second set of results we prove multiple query lower bounds on LP and SDP-solving in different input models and parameter regimes.

1.3 | Papers

This dissertation is based on the following papers:

[AG18] J. van Apeldoorn and S. Gribling. “Simon’s problem for linear functions”. 2018. arXiv: 1810.12030v2.

[AG19a] J. van Apeldoorn and A. Gilyén. “Improvements in Quantum SDP-Solving with Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). 2019, pp. 99:1–99:15. arXiv: 1804.05058v1.

[AG19b] J. van Apeldoorn and A. Gilyén. “Quantum algorithms for zero-sum games”. 2019. arXiv: 1904.03180v1.

[AGGW17] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. “Quantum SDP-Solvers: Better upper and lower bounds”. In: *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 403–414. arXiv: 1705.01843v3.

[AGGW18] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. “Convex optimization using quantum oracles”. 2018. Accepted to Quantum. arXiv: 1809.00643v3.

For all papers the co-authorship is shared equally.

Chapter 2

Preliminaries

After listing the basic notation and terminology used in this thesis, we will introduce the basic concepts from quantum computing that we will need. We then describe multiple algorithmic techniques that will be useful, which include amplitude amplification, phase estimation, quantum gradient computation and the framework of block-encodings. We finish this chapter with a discussion of some general concepts in convex optimization.

2.1 | Notation and terminology

Basic notation We use $\mathbf{i}^2 = -1$. We use square brackets $[\cdot]$ to denote the binary value corresponding to the truth-value of a statement, e.g. $[3 \le 5] = 1$. Square brackets with a positive integer in between them denote the set of positive integers up to and including that point: $[n] := \{1, 2, \ldots, n\}$. For sets $B_1, \ldots, B_n$ we use $\prod_{i=1}^n B_i$ to denote the Cartesian product $B_1 \times \cdots \times B_n$. For a set B we write $\mathscr{P}(B)$ for the power set of B. If b is a uniformly random element from B, then we write $b \in_R B$. We abbreviate "such that" with "s.t.". The non-negative and positive real numbers are denoted by $\mathbb{R}_{\ge 0}$ and $\mathbb{R}_{>0}$, respectively. We write $\mathbb{F}_p$ for the finite field with p elements. We let $\log(x)$ denote the base-2 logarithm. The notation $\mathrm{poly}(x, y)$ is used for an arbitrary polynomial in terms of x and y, and $\mathrm{polylog}(x, y)$ for an arbitrary polynomial in terms of $\log(x)$ and $\log(y)$. We write δ_{ij} for the Kronecker delta: $\delta_{ij} := [i = j]$. For a binary string $b \in \{0,1\}^n$ the notation $|b|$ denotes the Hamming weight (number of 1s) of b. For two binary strings $x, y \in \{0,1\}^n$ we write $x \oplus y$ for their element-wise XOR, and we write $\bar{x}$ for the element-wise negation of x. When we say that an algorithm has precision ε or calculates a quantity with error ε, then we mean *additive error*, unless stated otherwise. If an algorithm has error probability at most $1/3$ then we say it is a *bounded-error* algorithm, equivalently that the error probability is bounded. Unless otherwise stated the error probability can always be decreased to δ at the cost of an additional $\log(1/\delta)$ factor in the complexity.

Linear algebra When the dimension is clear from context we use e_i to denote the ith standard basis vector (the vector with a single 1 in position i and zeros everywhere else) and $\mathbf{1}$ to denote the all-ones vector. The matrix with a single 1 in position (i, j) and zeros everywhere else is denoted by E_{ij}. Let v, w be vectors and A, B be matrices. We write $v \ge w$ if v is entrywise greater or equal to w. The kernel of A is denoted by $\ker(A)$, and $\mathrm{rank}(A)$ denotes its rank. The Hadamard, or entrywise, product between two matrices is denoted by $A \circ B$. The tensor product between spaces, matrices or vectors is denoted by $\otimes$. The n-fold tensor product of A with itself is written as $A^{\otimes n}$. The direct sum of vector spaces, matrices or vectors is denoted by $\oplus$. We write $\mathrm{diag}(v)$ for the diagonal matrix with v on the diagonal. The inner product between real vectors $v, w \in \mathbb{R}^n$ is written as $\langle v, w \rangle := v^T w$. For a complex vector v or matrix M we write $v^\dagger$ or $M^\dagger$ to denote the conjugate transpose. A *Hilbert space* is an inner product space that is complete with respect to the metric coming from the inner product. A real matrix is called *positive semi-definite* (psd) if it is symmetric and all eigenvalues are non-negative, we write $X \succeq 0$ for a positive semi-definite matrix X. If H is a Hermitian matrix, then we write $\mathrm{Spec}(H)$ for its spectrum, i.e., the set of its eigenvalues. For a function $f : \mathbb{R} \to \mathbb{R}$ and Hermitian matrix H we write $f(H)$ for the matrix we get by applying f to the eigenvalues of H while keeping the eigenvectors the same, i.e.,

$$f(H) := U \begin{bmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{bmatrix} U^{-1} \text{ where } H = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} U^{-1}.$$

Geometry We write $\|x\|_p$ for the ℓ_p-norm of a vector x. If $p = 2$ then the subscript is sometimes omitted. We write $\|A\|$ for the *operator norm* of a matrix A, i.e., its largest singular value:

$$\|A\| := \max_{v \neq 0} \frac{\|Av\|}{\|v\|}.$$

We write $\|A\|_{tr}$ for the *trace norm* of a matrix A, i.e., the ℓ_1-norm of the vector of singular values. For a set $C \subseteq \mathbb{R}^n$ and $r \geq 0$ we write $B_p(C,r)$ for the set of points r-close to C in the ℓ_p-norm, i.e.,

$$B_p(C,r) := \left\{x \in \mathbb{R}^n : \exists y \in C \text{ s.t. } \left\|x - y\right\|_p \leq r\right\}.$$

When $C = \{x\}$ is a singleton set we abuse notation and write $B_p(x,r)$. We overload notation by letting

$$B_p(C,-r) := \left\{x \in \mathbb{R}^n : B_p(x,r) \subseteq C\right\}.$$

If $p = 2$ then the subscript is sometimes omitted. The set of interior points of C is written as $\mathrm{int}(C)$.

Functions Let f be a function. If f is a polynomial in terms of x, then $\deg_x(f)$ is the degree of f as a polynomial in the variable x. If f only has one variable, then the subscript is sometimes omitted. Let D, R be sets and let $A \subseteq D$. We say a function $f : A \to R$ is a *partial* function on D. A function $g : D \to R$ is said to *extend* f if and only if $g(x) = f(x)$ for all $x \in A$. We write $f \preceq g$ if g extends f. Let $C \subseteq \mathbb{R}^n$. A function $f : C \to \mathbb{R}$ is called *L-Lipschitz* if there exists a constant $L > 0$ such that

$$\left|f(y') - f(y)\right| \leq L\left\|y' - y\right\|_2 \text{ for all } y, y' \in C.$$

Big-O notation We define $\mathcal{O}(\cdot)$ as

$$f(x) = \mathcal{O}\big(g(x)\big) \iff \exists B \in \mathbb{R}, c \in \mathbb{R}_{>0} \text{ s.t. for all } x > B : f(x) \leq cg(x),$$

as usual. We define $\Omega(\cdot)$ as

$$f(x) = \Omega\big(g(x)\big) \iff g(x) = \mathcal{O}\big(f(x)\big)$$

and

$$f(x) = \Theta\big(g(x)\big) \iff f(x) = \mathcal{O}\big(g(x)\big) \text{ and } f(x) = \Omega\big(g(x)\big).$$

Furthermore we use the following definition of $\widetilde{\mathcal{O}}(\cdot)$:

$$\widetilde{\mathcal{O}}\big(f(x)\big) := \mathcal{O}\big(f(x) \cdot \mathrm{polylog}(f(x))\big).$$

If a function depends polylogarithmically on additional variables that we also hide using the $\widetilde{\mathcal{O}}(\cdot)$ notation, then we write these as a subscript to the $\widetilde{\mathcal{O}}(\cdot)$:

$$\widetilde{\mathcal{O}}_{a,b}\big(f(x)\big) := \mathcal{O}\big(f(x) \cdot \mathrm{polylog}(a, b, f(x))\big).$$

We define $\widetilde{\Omega}(\cdot)$ and $\widetilde{\Theta}(\cdot)$ in a similar way as above.

2.2 | Basics of quantum computing

Below we give an overview of some of the basic concepts in quantum computing that will be used later. For a more complete introduction see for example the textbook by Nielsen and Chuang [NC00] or the lecture notes by de Wolf [Wol19].

2.2.1 | Quantum states, Dirac notation and qubits

In this thesis we consider systems that have a finite number N of *classical* states. These classical states often are denoted by $|0\rangle, \ldots, |N-1\rangle$. In the classical world these states are mutually exclusive. The *quantum* analogue of such a classical system can, however, be in a *superposition* of states where each classical state $|i\rangle$ is assigned a complex number α_i called its *amplitude*. We associate the quantum system with an N-dimensional *Hilbert space* and the N classical states with the N standard basis vectors of that space (also called the *computational basis*). A *pure state* of the quantum system is a unit vector in this Hilbert space, which can be written as

$$\left|\phi\right\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{N-1} \end{bmatrix}$$

where $\sum_{i=0}^{N-1} |\alpha_i|^2 = 1$. The part of the amplitude corresponding to the argument of the complex number, i.e., the part of the form $e^{\mathbf{i}\theta}$ is called a *phase*.

The notation $\left|\phi\right\rangle$ is called a *ket* and comes from a system called *Dirac notation*. For a column vector $\left|\phi\right\rangle$ we write $\left\langle\phi\right|$ (called a *bra*) for the row vector which is its complex conjugate. The inner product between states can then be written as $\left\langle\phi\middle|\psi\right\rangle$.

When two quantum systems with Hilbert spaces $\mathscr{H}_1$ and $\mathscr{H}_2$ are considered together then the state of the full system lives inside the space corresponding to the tensor product of the two spaces: $\mathscr{H}_1 \otimes \mathscr{H}_2$. The basis states for this new space correspond to the tensor products of the basis states of the separate spaces

$$|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |0\rangle \otimes |2\rangle, \ldots, |1\rangle \otimes |0\rangle, \ldots$$

For ease of writing we also write $\left|\phi\right\rangle\left|\psi\right\rangle$ for the tensor product between states $\left|\phi\right\rangle$ and $\left|\psi\right\rangle$, and even $\left|i,j\right\rangle$ for the tensor product between classical basis states $|i\rangle$ and $\left|j\right\rangle$. A state in the combined space that can be written as a tensor product $\left|\phi_1\right\rangle \otimes \left|\phi_2\right\rangle$ is called *separable*. A state that is not separable is called *entangled*.

The simplest non-trivial classical system is a *bit*. A bit has two states, $|0\rangle$ and $|1\rangle$. The corresponding quantum system is called a *qubit*. As classical bits can be combined, so can qubits be combined via tensor products to create larger systems. We often group qubits in *registers*, a k-qubit register can be in any of 2^k classical states. We sometimes write $|0^n\rangle$, or simply $|0\rangle$, for the n-qubit zero state.

2.2.2 | Measurements

Since we ourselves live in the classical world we need some procedure to gain classical information from a quantum state. This is done via *measurements*. A measurement does not give us a classical description of the vector corresponding to a state, but is an inherently probabilistic process. By measuring we gain some discrete knowledge about the state. After the measurement the state *collapses* to the part of the original state that agrees with this information (and is re-normalized).

The simplest type of measurement is a measurement in the computational basis. Given a state $|\phi\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$ a measurement in the computational basis yields outcome i with probability $|\alpha_i|^2$. After the measurement the state changes to $|i\rangle$ and repeating the measurement will yield the same result. If only part of a state is measured then the probability of an outcome is equal to the squared norm of the part of the state consistent with that outcome. For example, consider the state $|\phi\rangle = \frac{1}{2}|0\rangle|0\rangle + \frac{1}{2}|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle$. If the first register is measured, then the probability of getting outcome 0 is $|1/2|^2 + |1/2|^2 = 1/2$. If we get outcome 0, then the state becomes $\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|0\rangle|1\rangle$. Note that the normalization ensures that the probabilities sum to 1.

Measurements in the computational basis suffice for all quantum algorithms, however, for analysis it is sometimes useful to consider the more general *projective measurements*. A projective measurement corresponds with a set of projectors $P_0, \dots, P_{k-1}$ with the property that $\sum_{i=0}^{k-1} P_i = I$. The probability of getting outcome i when measuring the state $|\phi\rangle$ is equal to $\left\| P_i |\phi\rangle \right\|^2$. After getting outcome i the state collapses to $\frac{P_i|\phi\rangle}{\|P_i|\phi\rangle\|}$. The projective measurement $\{E_{ii} : i \in \{0, \dots, N-1\}\}$ corresponds to a measurement in the computational basis.

Finally, the most general type of measurement is a *positive-operator valued measurement* (POVM). A POVM is given by a collection of positive semidefinite matrices $M_0, \dots, M_{k-1}$ with the property that $\sum_{i=0}^{k-1} M_i = I$. The probability of getting outcome i when measuring $|\phi\rangle$ is equal to $\langle\phi|M|\phi\rangle = \mathrm{Tr}(M_i|\phi\rangle\langle\phi|)$.

2.2.3 | Unitaries and gates

Apart from measurements, which are non-linear and destructive, we can also apply certain linear maps to transform states. Since a state should always remain normalized, any linear map that we may apply to a state should maintain this property. This implies that we are limited to *unitary* maps, i.e., those with eigenvalues on the complex unit circle. In fact, the unitaries are exactly the maps that can be applied to a quantum state. For a unitary U the inverse U^{-1} is equal to the conjugate transpose $U^\dagger$, which is also unitary. This implies that quantum processes, apart from measurements, are reversible. Unitaries do not change the inner product between states. Finally, it suffices to specify their behavior on the computational basis states, since the behavior on superpositions follows by linearity.

In physics the evolution of a system is often described by a *Hamiltonian*, which is a

Hermitian matrix. The Hamiltonian H describes the *energy* of a quantum system, with the expectation value of the energy being $\langle\phi|H|\phi\rangle$. If we start in a state $|\phi\rangle$ and evolve under H for time t then the state becomes $e^{\mathbf{i}tH}|\phi\rangle$. Since all eigenvalues of H are real the matrix $e^{\mathbf{i}tH}$ is a unitary. The reverse is also true, if U is a unitary then there exists a Hamiltonian H such that $e^{\mathbf{i}H} = U$.

Although all unitaries can in principle be applied to a state, it is not always clear how to do so efficiently. Similar to the classical concept of *gates* that apply a function (often NOT, OR, or AND) to a small number of bits, we also have a notion of a *quantum gate* that applies to a small number of qubits. Larger unitaries are then implemented as a sequence of gates, with the gate count being the main complexity measure. The following single-qubit gates are of special interest:

$$I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad X := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Z := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad H := \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

I is the identity and does not change the state, X corresponds to a classical NOT gate (it flips the qubit it is applied to), and Z applies a phase of -1 to $|1\rangle$. Finally H is called the *Hadamard transform* and transforms a basis state into a superposition:

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) =: |+\rangle \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =: |-\rangle \end{aligned}$$

It can easily be seen that the Hadamard transform is its own inverse:

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

and similar for $|1\rangle$. The cancelling of the two $|1\rangle$ parts of the state is called *interference*. Note that a Z gate switches the $|+\rangle$ and $|-\rangle$ states, just like an X gate does for the $|0\rangle$ and $|1\rangle$ states. In fact $X = HZH$ and $Z = HXH$, since H corresponds with the basis transform that maps the $|0\rangle, |1\rangle$ basis to the $|+\rangle, |-\rangle$ basis.

Although all these gates act only on one qubit, they can also be applied to a single qubit of a larger quantum system. In that case we actually apply the unitary that is the tensor product of a lot of identity gates and the single-qubit gate we want to apply.

Apart from single-qubit gates we also need a two-qubit gate, since otherwise the qubits could not influence each other. The simplest two-qubit gate is the *controlled-NOT* ($CNOT$) gate. This gate applies an X gate to the second qubit when the first qubit is in the state $|1\rangle$. As a matrix this can be written as

$$CNOT := |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = I \oplus X = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

In fact it can be shown[1] that any unitary can be implemented using only one-qubit gates and $CNOT$ gates, although this might require up to $\mathcal{O}(4^n)$ gates for an n-qubit unitary. When discussing the number of gates used to implement a quantum algorithm we will count the number of two-qubit gates used. Sometimes it is easier to implement a unitary when extra qubits, called *ancilla* qubits, are used. If this is done then we often require the ancilla qubits to be returned to their original state in the end, since otherwise unwanted entanglement could be created between the ancillas and the main register. Any classical algorithm consisting of T classical gates can also be implemented with $\Theta(T)$ quantum gates (although more space might be needed to store intermediate answers that will be used to make the circuit reversible).

Like the $CNOT$ gate is a controlled version of the X gate, other unitaries can also have controlled versions. In general a controlled version of the unitary U is

$$CU = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U.$$

Controlled gates allow us to delay measurements until the end of an algorithm, since we can control the gates we would normally apply after a measurement on the fact that the measured register is in a particular state.

2.2.4 | Oracles and QRAM

While the output of a quantum algorithms is given by a measurement, we have not yet discussed the input. For an input string $x \in \{0,1\}^n$ a classical algorithm would make calls to the memory to retrieve the bits of x. Similarly, a quantum algorithm can be given access to a unitary, called an *oracle*, that encodes the string x. The simplest encoding is called a *standard oracle* and acts as

$$O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$$

for $i \in [n]$ and $b \in \{0,1\}$. Similarly for a function f that outputs values in $\{0,1\}^k$, a standard oracle for that function is a unitary that maps $\left|y\right\rangle\left|b\right\rangle$ to $\left|y\right\rangle\left|b \oplus f(y)\right\rangle$ for $y \in \dim\left(f\right)$ and $b \in \{0,1\}^k$.

There is also a different type of oracle called a *phase oracle*. For a string $x \in \{0,1\}^n$ a phase oracle acts as

$$O_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$$

for $i \in [n]$. A phase oracle for a string x can be constructed from a standard oracle by applying the standard oracle O_x with $|-\rangle$ in the second register:

$$O_x|i\rangle|-\rangle = \frac{O_x|i\rangle|0\rangle - O_x|i\rangle|1\rangle}{\sqrt{2}} = \frac{|i\rangle|x_i\rangle - |i\rangle|\bar{x}_i\rangle}{\sqrt{2}} = (-1)^{x_i}\frac{|i\rangle|0\rangle - |i\rangle|1\rangle}{\sqrt{2}} = (-1)^{x_i}|i\rangle|-\rangle,$$

[1] See for example [NC00, Sec. 4.5].

which is exactly the result of a query to $O_{x,\pm}$. In fact, we can implement a conditional phase oracle, since applying the standard oracle with $|+\rangle$ in the second register leaves the state the same.

The standard quantum oracle described above models problems where there is a single correct answer to a query. When there are multiple good answers (for instance, different good approximations to the correct value) and the oracle is only required to give a correct answer with high probability, then we will work with the more liberal notion of *relational* quantum oracles.

Definition 2.1 : Relational quantum oracle *Let $\mathscr{F}: X \to \mathscr{P}(Y)$ be a function, such that for each $x \in X$ the subset $\mathscr{F}(x) \subseteq Y$ is the set of valid answers when x is queried. A relational quantum oracle for $\mathscr{F}$ which answers queries with success probability $\geq 1-\rho$, is a unitary that for all $x \in X$ maps*

$$U\colon |x\rangle|0\rangle|0\rangle \mapsto \sum_{y\in Y} \alpha_{x,y}\big|x\big\rangle\big|y\big\rangle\big|\psi_{x,y}\big\rangle,$$

where $\big|\psi_{x,y}\big\rangle$ denotes some normalized quantum state and $\sum_{y\in\mathscr{F}(x)}\big|\alpha_{x,y}\big|^2 \geq 1-\rho$. Thus measuring the second register of $U|x\rangle|0\rangle|0\rangle$ in the computational basis gives a valid answer to the query x with probability at least $1-\rho$.

This definition is very natural for cases where the oracle is implemented by a quantum algorithm that produces a valid answer with probability $\geq 1-\rho$.

For standard oracles we can assume that we can apply their inverse as well, since they are their own inverse, and we may assume that we can apply them controlled as well, since they act like identity when the second register is in a uniform superposition. However, for other oracles this is not always the case. When we additionally assume access to the inverse of an oracle or a controlled version of an oracle then this will always be stated. For oracles coming from quantum algorithms, and for quantum algorithms in general, it is natural to assume both since we can always invert the quantum algorithm by applying the inverses of the gates in inverse order and we can make the algorithm controlled by making all the gates controlled.[2]

Finally we return to the classical notion of random access memory, or RAM. In classical works it is often assumed that a read or write operation to the memory takes $\mathscr{O}(1)$ time, or at most $\widetilde{\mathscr{O}}_K(1)$ time for a memory of size K. However, implementing a RAM using gates takes $\Omega(K)$ gates. This discrepancy comes from the fact that a classical RAM can be implemented in a highly parallelized manner. Whether an equivalent model for quantum computing is reasonable is an ongoing debate. To make the comparison with classical algorithms fair we often assume access to some form of quantum memory. There are two forms to distinguish. The first is *quantum-read/classical-write* RAM (QCRAM), which allows us to store classical data coming from measurements and allows the quantum algorithm to make oracle calls to this data. The second form is a *full*

[2] Making all the gates controlled does increase the number of qubits per gate, but these larger gates can be implemented with a constant number of smaller gates.

QRAM, which allows the quantum computer to swap a quantum register it is working on with a quantum register stored in memory as follows:

$$QRAM : |i, x, r_1, \dots, r_K\rangle \mapsto |i, r_i, r_1, \dots, r_{i-1}, x, r_{i+1}, \dots, r_K\rangle,$$

where the registers that initially contain $r_1, \dots, r_K$ are only accessible through this gate. Either of these quantum RAMs can be implemented using $\widetilde{\mathcal{O}}(K)$ two-qubit gates, where K is the size of the memory.

2.2.5 | Mixed states

In some settings it is natural to consider quantum states that come from a classical probability distribution. An example of this is when a state is measured and the measurement result is ignored, or when we only have access to a few of the qubits of an entangled state. A *mixed state* is a probability distribution over pure states. A mixed state is represented by a *density matrix*, also called a *density operator*, which is a trace-normalized positive semidefinite matrix. A pure state $|\phi\rangle$ corresponds to the rank-one mixed state $|\phi\rangle\langle\phi|$. If for $i \in [r]$ we have the state $|\phi_i\rangle$ with probability p_i, then the corresponding density matrix is the linear combination of the individual rank-one density matrices:

$$\rho = \sum_{i=1}^{r} p_i |\phi_i\rangle\langle\phi_i|,$$

where $p_i \geq 0$ and $\sum_{i=1}^{r} p_i = 1$.

For every density matrix there is always a decomposition into pure states, although this decomposition is not always unique. For an N-dimensional density operator such a decomposition requires at most N pure states. Unitaries act on density operators by conjugation: ρ is mapped to $U\rho U^\dagger$ under U. POVMs naturally extend to mixed states as well by averaging the probability over the set of pure states and using the linearity of the trace. Let $\{M_i\}$ be a POVM:

$$\begin{aligned}
\Pr\left[j \text{ is measured in state } \rho\right] &= \sum_{i=1}^{r} p_i \Pr\left[j \text{ is measured in state } |\phi_i\rangle\right] \\
&= \sum_{i=1}^{r} p_i \mathrm{Tr}\left(M_j |\phi_i\rangle\langle\phi_i|\right) \\
&= \mathrm{Tr}\left(M_j \sum_{i=1}^{r} p_i |\phi_i\rangle\langle\phi_i|\right) \\
&= \mathrm{Tr}\left(M_j \rho\right).
\end{aligned}$$

The idea of measuring one register and ignoring the result gives rise to the definition of the *partial trace*. Let

$$M = M_A \otimes M_B$$

be a matrix on $\mathscr{H}_A \otimes \mathscr{H}_B$, then the partial trace over B is

$$\mathrm{Tr}_B(M) = M_A \mathrm{Tr}(M_B),$$

which is a matrix on $\mathscr{H}_A$. We define the partial trace for matrices that are not a tensor product by linearly extending this definition. We say B is *traced out*. A pure state $|\phi\rangle \in \mathscr{H}_A \otimes \mathscr{H}_B$ is called a *purification* of a mixed state ρ on $\mathscr{H}_A$ if tracing out B from $|\phi\rangle\langle\phi|$ results in ρ.

Sometimes we care only about a certain part of a mixture, for which the corresponding matrix can be of trace less than or equal to one. This leads to the following definition.

Definition 2.2 : Subnormalized density operators & Purification *A* subnormalized density operator ϱ *is a psd matrix of trace at most* 1*. A* purification *of a subnormalized density operator* ϱ *is a* 3*-register pure state such that tracing out the third register and projecting on the subspace where the second register is* $|0\rangle$ *yields* ϱ*.*

We write "ϱ" and "ς" for subnormalized density operators to distinguish them from normalized density operators, for which we write "ρ" and "σ".

2.2.6 | A first example of an algorithm: the Deutsch-Jozsa algorithm

Consider the following problem:

> *Let n be a positive integer and $N := 2^n$. Let $x \in \{0,1\}^N$ be such that*
>
> - *either $|x| = N/2$, in which case we call x balanced,*
> - *or $|x| \in \{0, N\}$, in which case we call x constant.*
>
> *Decide whether x is balanced or constant.*

Clearly a classical deterministic algorithm would have to read at least $N/2+1$ bits of x to solve the problem, since after seeing $N/2$ bits that are 1, we still cannot be sure whether the input is balanced or constant. However, Deutsch and Jozsa showed [DJ92] that there exists a quantum algorithm that solves this problem with certainty using only a single query! First note that the Hadamard transform can be written as

$$H|b\rangle = \frac{1}{\sqrt{2}} \sum_{j \in \{0,1\}} (-1)^{b \cdot j} |j\rangle.$$

This means that applying a Hadamard gate to each of n qubits acts as

$$H^{\otimes n}|i\rangle = \frac{1}{\sqrt{N}} \sum_{j \in \{0,1\}^n} (-1)^{\langle i,j\rangle} |j\rangle,$$

where $\langle i,j\rangle$ is the inner product between the binary representations of i and j. Now, consider the following quantum algorithm that uses only one query:

1. Start in the $|0\rangle$ state on n qubits.

2. Apply the Hadamard transform to each of the qubits, creating a *uniform superposition*
$$H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle.$$

3. Apply a phase query $O_{x,\pm}$ to get the state
$$\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(-1)^{x_i}|i\rangle.$$

4. Apply the Hadamard transform again to all the qubits:
$$\frac{1}{N}\sum_{i=0}^{N-1}(-1)^{x_i}\sum_{j=0}^{N-1}(-1)^{\langle i,j\rangle}|j\rangle = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}(-1)^{x_i+\langle i,j\rangle}|j\rangle.$$

5. Measure the state in the computational basis and output "constant" if and only if the outcome is the all-zero state.

The amplitude of the all-zero state before the measurement is
$$\frac{1}{N}\sum_{i=0}^{N-1}(-1)^{x_i+\langle i,0\rangle} = \frac{1}{N}\sum_{i=0}^{N-1}(-1)^{x_i} = \begin{cases} 0 & \text{if } x \text{ is balanced} \\ 1 & \text{if } x \text{ is all zeros} \\ -1 & \text{if } x \text{ is all ones.} \end{cases}$$

Hence measuring will give the all-zero state as an outcome if and only if x is constant.

2.3 | Useful quantum algorithmic techniques

Below we will give an overview of some of the algorithmic techniques used in the first part of this thesis.

2.3.1 | Amplitude amplification and Grover search

In classical randomized algorithms there are often outcomes that we deem "good" and outcomes that we deem "bad". If there is a probability p of obtaining a good outcome from a randomized algorithm, then we can repeat the algorithm $1/p$ times to obtain a constant success probability. In a similar fashion, for a state coming from a quantum algorithm we can *amplify* the amplitude on a certain part of the state, without measuring and retrying. In fact, we gain a quadratic speedup:

Lemma 2.3 : Amplitude amplification [BHMT02] *Let U be a unitary that maps $|0\rangle$ to the state*

$$\left|\psi\right\rangle = \cos(\theta)\left|\phi_0\right\rangle\left|0\right\rangle + \sin(\theta)\left|\phi_1\right\rangle\left|1\right\rangle$$

for some (small) angle θ. Let $G := U(2|0\rangle\langle 0| - I)U^\dagger(I \otimes Z)$ be the unitary that adds a phase of -1 first to the states ending in a $|1\rangle$ and then to the states orthogonal to $\left|\psi\right\rangle$. Then G is a rotation by angle 2θ in the subspace spanned by $\left|\phi_0\right\rangle\left|0\right\rangle$ and $\left|\phi_1\right\rangle\left|1\right\rangle$. In particular, after k applications of G to $\left|\psi\right\rangle$ we obtain the state

$$\left|\psi'\right\rangle = \cos((2k+1)\theta)\left|\phi_0\right\rangle\left|0\right\rangle + \sin((2k+1)\theta)\left|\phi_1\right\rangle\left|1\right\rangle.$$

If we would like to obtain the "good" state $\left|\phi_1\right\rangle$ and would not like to obtain the "bad" state $\left|\phi_0\right\rangle$, then we could do so with high probability by picking k such that $(2k+1)\theta \approx \pi/2$. If we would instead measure $\left|\psi\right\rangle$ directly then this would give us a probability $p = \sin(\theta)^2$ of finding the good state and hence a classical prepare-and-retry strategy would need an expected number of $1/p$ tries (and hence applications of U). However, using amplitude amplification we see that $k = \left\lceil\left(\frac{\pi}{2\theta} - 1\right)/2\right\rceil = \mathcal{O}\left(\frac{1}{\sqrt{p}}\right)$ applications of U and $U^\dagger$ suffice.

Example: Grover search Consider the following problem, called the *search problem*:

> *Let $x \in \{0,1\}^n$. Find an i such that $x_i = 1$ or conclude that no such i exists.*

Assume that the number of 1s in x is known and call this number t. Classically $\Omega(n/t)$ randomized queries are needed to solve this problem.

Let U be a unitary that prepares a uniform superposition $\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle$ (starting from $|0\rangle$), and queries a binary oracle for x to get the state

$$\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle|x_i\rangle = \sqrt{\frac{t}{n}}\left(\frac{1}{\sqrt{t}}\sum_{i:x_i=1}|i\rangle\right)|1\rangle + \sqrt{\frac{n-t}{n}}\left(\frac{1}{\sqrt{n-t}}\sum_{i:x_i=0}|i\rangle\right)|0\rangle.$$

Using $k = \mathcal{O}\left(\sqrt{n/t}\right)$ calls to U and $U^\dagger$ (and hence so many queries) we can with probability at least $2/3$ obtain the state

$$\frac{1}{\sqrt{t}}\sum_{i:x_i=1}|i\rangle.$$

Measuring this state results in a solution to the search problem.

This search algorithm was first given by Grover in 1996 [Gro96]. In fact, the more general algorithm for amplitude amplification was inspired by Grover's algorithm.

2.3.2 | Phase estimation

Sometimes we will have a clear description and understanding of the unitaries that we apply to quantum states, but it is also common that we do not have full knowledge about them. In fact, gaining knowledge about a certain unitary might exactly be the problem we are trying to solve, or at least help us solve the problem. A natural question to ask is what the *eigenvalues* (or the *phases* of the eigenvalues) and eigenvectors of a unitary are.

Lemma 2.4 : Phase estimation [Kit95][CEMM98] *Let U be a unitary and $|\psi\rangle$ be an eigenvector of U with eigenvalue $e^{\mathbf{i}2\pi\theta}$, for $-1/2 < \theta \leq 1/2$. Let $k \in \mathbb{N}$ and let V be a unitary that acts on two registers as $V|j\rangle|\phi\rangle = |j\rangle U^j|\phi\rangle$ for all $j \leq k$. Then there is a bounded-error quantum algorithm that on input $|\psi\rangle$ approximates θ up to additive error ε using a single application of V for $k = \mathcal{O}\left(\frac{1}{\varepsilon}\right)$. If θ can be described exactly with t bits of precision then the algorithm works with probability* 1*, $k = 2^t$ suffices, and the algorithm always outputs θ.*

Phase estimation uses a quantum algorithm called the *quantum Fourier transform* (QFT). This is a unitary that acts on an N-dimensional Hilbert space as

$$QFT_N|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\mathbf{i}2\pi jk/N}|k\rangle.$$

If $N = 2^n$ then this unitary (and its inverse) can be implemented using $\mathcal{O}(n^2)$ two-qubit gates, or approximated very closely using $\mathcal{O}(n\log(n))$ gates.

Assume that $\theta = \frac{j}{N}$. Phase estimation can be implemented by setting up a uniform superposition over $0,\ldots,N-1$, querying V to obtain phases and then applying the *inverse quantum Fourier transform*:

$$\begin{aligned}
|0\rangle|\psi\rangle &\mapsto \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}|k\rangle|\psi\rangle \\
&\mapsto \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}|k\rangle U^k|\psi\rangle \\
&= \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}|k\rangle\left(e^{\mathbf{i}2\pi j/N}\right)^k|\psi\rangle \\
&= \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{\mathbf{i}2\pi kj/N}|k\rangle|\psi\rangle \\
&\mapsto |j\rangle|\psi\rangle.
\end{aligned}$$

Example: Amplitude estimation [BHMT02] Let U, $|\psi\rangle$ and G be as in Lemma 2.3. One of the problems with amplitude amplification as presented in the last section is that we may not know θ, and hence we may not know how many times we would like to apply G. Assume that $0 \leq \theta < \pi$, can we compute θ?

Note that G is a rotation by angle 2θ in the 2-dimensional subspace spanned by $|\phi_0\rangle|0\rangle$ and $|\phi_1\rangle|1\rangle$, hence the eigenvalues of G are $e^{\mathbf{i}2\theta}$ and $e^{-\mathbf{i}2\theta}$. We also know that $|\psi\rangle$ is some linear combination of the eigenvectors of G, i.e.,

$$|\psi\rangle = \beta_0|\zeta_0\rangle + \beta_1|\zeta_1\rangle$$

for $|\zeta_0\rangle$ and $|\zeta_1\rangle$ the eigenvectors of G in the 2-dimensional subspace. Assume for simplicity that θ/π (and hence $-\theta/\pi$) can be described by k bits exactly, then phase estimation on G with input $|\psi\rangle$ gives us

$$\beta_0|\zeta_0\rangle|\theta/\pi\rangle + \beta_1|\zeta_1\rangle|-\theta/\pi\rangle.$$

Clearly measuring the second register suffices to learn θ. Note that we learned θ, and hence $\sin(\theta)$, up to additive error ε using $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ applications of G (and hence of U and $U^\dagger$). If we would simply have measured $|\psi\rangle$ many times and used a classical estimate then this would have required $\Omega\left(\frac{1}{\varepsilon^2}\right)$ preparations of $|\psi\rangle$.

2.3.3 | Quantum gradient computation

Many optimization algorithms use the gradient of a function in order to minimize that function. In some cases we have an explicit description of the gradient of a function, but often we do not. A natural question is whether it is possible to approximate the gradient using only black-box access to the function. In general if $f:\mathbb{R}^n\to\mathbb{R}$ then a classical algorithm will need at least $\widetilde{\Theta}_\delta(n)$ queries to a black-box oracle for f in order to approximate the gradient up to error δ in the ℓ_∞-norm. In 2005 Jordan proposed a quantum algorithm that can do better when f is close to a linear function [Jor05]. We will need the following definition:

Definition 2.5 : Hyper-grid *For $k\in\mathbb{N}$ we define the following discretization of the interval $(-1/2,1/2)$:*

$$G_k := \left\{\frac{j}{2^k} - \frac{1}{2} + 2^{-k-1} : j \in \left\{0,\ldots,2^k-1\right\}\right\} \subset (-1/2,1/2).$$

Similarly we define the n-dimensional hyper-grid $G_k^n := \prod_{i=1}^n G_k$.

Note that an element of G_k^n can be represented using $n\times k$ (qu)bits. Basically, Jordan's algorithm sets up a uniform superposition over all grid points, applies a “phase query” to f, and then an inverse quantum Fourier transform over each coordinate. If $f(x)\approx c+\langle g,x\rangle$ then the phase of $e^{\mathbf{i}f(x)}$ coming from a query is approximately $e^{\mathbf{i}c}\prod_{i=1}^n e^{\mathbf{i}g_i x_i}$, where $e^{\mathbf{i}c}$ is a global phase that we can ignore. The phases $e^{\mathbf{i}g_i x_i} = \left(e^{\mathbf{i}g_i}\right)^{x_i}$ can be distributed over the n registers (one register for each G_k). Jordan's algorithm therefore can be viewed as n simultaneous applications of phase estimation, where one phase query to f can be used to run all applications of phase estimation simultaneously. A careful analysis of Jordan's algorithm was made by Gilyén, Arunachalam and Wiebe [GAW19]. Their result can be combined with a simple reduction from a standard oracle to a phase oracle [AGGW18] to obtain the following lemma.

Lemma 2.6 : [GAW19, Thm. 14, Thm. 21][AGGW18, Cor. 15] *Let $\delta,B,r\in\mathbb{R}_{>0}$, $c\in\mathbb{R}$, $\rho\in(0,1/3]$. Let $x_0,g\in\mathbb{R}^n$ with $\|g\|_\infty\le\frac{B}{r}$. Let $m:=\lceil\log_2(\frac{B}{28\pi\delta})\rceil$ and suppose $f:\left(x_0+rG_m^n\right)\to\mathbb{R}$ is such that*

$$\left|f(x_0+rx)-\langle g,rx\rangle-c\right|\le\delta \tag{2.1}$$

for 99.9% of the points $x \in G_m^n$. Then we can compute a vector $\tilde{g} \in \mathbb{R}^n$ such that

$$\Pr\left[\left\|\tilde{g} - g\right\|_\infty > \frac{16 \cdot 42\pi\delta}{r}\right] \leq \rho,$$

with either of the following complexities:

- *Let $\tilde{f}$ be such that $\tilde{f}(z)$ can be described in $\mathcal{O}\left(\log(B/\delta)\right)$ bits and that $\left|\tilde{f}(z) - f(z)\right| \leq \delta$ for all $z \in \left(x_0 + rG_m^n\right)$. Then $\mathcal{O}\left(\log\left(\frac{n}{\rho}\right)\right)$ queries to a standard binary quantum oracle U for $\tilde{f}$ on G_m^n and to the inverse $U^\dagger$, and gate complexity*

  $$\mathcal{O}\left(n\log\left(\frac{n}{\rho}\right)\log\left(\frac{B}{\delta}\right)\log\log\left(\frac{n}{\rho}\right)\log\log\left(\frac{B}{\delta}\right)\right)$$

  suffice.

- *Alternatively, $\mathcal{O}\left(\log\left(\frac{n}{\rho}\right)\right)$ queries to a phase oracle O for f on G_m^n acting as $O|x\rangle = e^{\mathbf{i}\frac{1}{42\delta}f(x_0+rx)}|x\rangle$, and gate complexity*

  $$\mathcal{O}\left(n\log\left(\frac{n}{\rho}\right)\log\left(\frac{B}{\delta}\right)\log\log\left(\frac{n}{\rho}\right)\log\log\left(\frac{B}{\delta}\right)\right)$$

  suffice.

- *Or, if $\operatorname{range}(f) = [0,1]$, then $\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{1}{\delta}\right)\log\left(\frac{n}{\rho}\right)\right)$ queries to a probability oracle acting as*

  $$O_p|x\rangle|0\rangle|0\rangle = |x\rangle\left(\sqrt{f(x)}|0\rangle|\psi_0\rangle + \sqrt{1-f(x)^2}|1\rangle|\psi_1\rangle\right),$$

  and gate complexity

  $$\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{1}{\delta}\right)\log\left(\frac{n}{\rho}\right) + n\log\left(\frac{n}{\rho}\right)\log\left(\frac{B}{\delta}\right)\log\log\left(\frac{n}{\rho}\right)\log\log\left(\frac{B}{\delta}\right)\right)$$

  suffice.

This approach for binary oracles can in fact also be extended to relational quantum oracles, see the appendix of [AGGW18] for details.

Example: Estimating a whole distribution Let p be a probability distribution over n elements and let U be a unitary that creates this distribution as a quantum state:

$$U|0\rangle = \sum_{i=1}^{n} \sqrt{p_i}|i\rangle|\psi_i\rangle,$$

where the $|\psi_i\rangle$ are arbitrary states. Amplitude estimation can be used to estimate one of the p_i up to precision δ using only $\mathcal{O}\left(\frac{1}{\delta}\right)$ applications of U and $U^\dagger$. However, if we would want to estimate *all* the p_i's then simply repeating this procedure would give a rather bad bound of $\mathcal{O}\left(\frac{n}{\delta}\right)$ queries.

Now consider the function $f(x) = \langle p, x\rangle$. Clearly $\nabla_x f = p$ and hence a δ-precise estimate of the gradient in ℓ_∞-norm would give the desired result. We can implement a probability oracle for $f(x)$ when $x \in B_\infty\left(\frac{1}{2}\mathbf{1}, \frac{1}{2}\right)$ by querying U and then rotating depending on x:

$$\begin{aligned}
|x\rangle|0\rangle|0\rangle &\mapsto |x\rangle \sum_{i=1}^{n} \sqrt{p_i}|i\rangle|0\rangle \\
&\mapsto |x\rangle \sum_{i=1}^{n} \sqrt{p_i}|i\rangle\left(\sqrt{x_i}|0\rangle + \sqrt{1-x_i}|1\rangle\right) \\
&= |x\rangle\left(\sum_{i=1}^{n} \sqrt{p_i x_i}|i\rangle|0\rangle + \sum_{i=1}^{n} \sqrt{p_i - p_i x_i}|i\rangle|1\rangle\right) \\
&= |x\rangle\left(\sqrt{\langle p, x\rangle}\left(\sum_{i=1}^{n} \sqrt{\frac{p_i x_i}{\langle p, x\rangle}}|i\rangle\right)|0\rangle + (\ldots)|1\rangle\right).
\end{aligned}$$

Hence, by the last bullet of Lemma 2.6, $\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{n}{\rho}\right)\log\left(\frac{1}{\delta}\right)\right)$ queries suffice to estimate p up to ℓ_∞-norm error δ with error probability ρ.

2.3.4 | Block-encodings

Many problems in computer science and machine learning involve linear algebra. Examples include solving a linear system, solving the least square problem, and many forms of image transformation. A natural question to ask is whether we can use the inherent linear nature of quantum mechanics to construct fast quantum algorithms for these tasks. It seems natural to encode vectors as the amplitudes of quantum states and then transform these using a quantum algorithm. One problem we need to overcome for such an approach is the limitation that we can only apply unitaries to quantum states, and we cannot apply general matrices. We can however embed a general matrix in one part of a unitary. This leads to the following definition:

Definition 2.7 : Block-encoding [LC16; GSLW19] *Suppose that A is a 2^w-dimensional matrix, $\alpha \in \mathbb{R}_{>0}$, $\varepsilon \in \mathbb{R}_{>0}$ and $k \in \mathbb{N}$, if for an $(a+w)$-qubit unitary U we have*

$$\left\| A - \alpha\left(\langle 0|^{\otimes a} \otimes I\right) U \left(|0\rangle^{\otimes a} \otimes I\right)\right\| \leq \varepsilon,$$

then we call U an (α, a, ε)-block-encoding *of A.*

Roughly speaking this means that A is represented by a unitary of the form

$$U \approx \begin{bmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix}$$

where each "$\cdot$" can be an arbitrary matrix.

A block-encoding allows us to apply A to a state in a certain way, even if A is not a unitary. In particular, if we would like to apply A to $|\psi\rangle$ then we could add an ancilla qubit that starts in the zero state, and apply U to get

$$U|\psi\rangle|0\rangle = \frac{1}{\alpha}A|\psi\rangle|0\rangle + (\dots)|\perp\rangle$$

where $\langle 0|\perp\rangle = 0$. Hence by post-selecting on the $|0\rangle$ state being in the second register, or by amplifying this part of the state, we can obtain the state

$$\frac{A|\psi\rangle}{\|A|\psi\rangle\|}.$$

Since unitaries are also matrices, we will sometimes call a unitary a $(1,a,\varepsilon)$-block-encoding of another unitary if it uses a ancillary qubits and is an ε-approximation in the operator norm. Note that every unitary is a $(1,0,0)$-block-encoding of itself.

Of course this all raises the question whether block-encodings can easily be implemented. As it turns out many natural input models for matrices can be used to implement block-encodings. For example when a matrix is given via sparse matrix access, via Hamiltonian simulation, as a density operator or as a measurement then we can efficiently implement a block-encoding of the matrix as well. We will discuss this in more depth in Chapter 4.

Block-encodings can be multiplied in the natural way, but they can also be added together if we have access to a controlled version of the block-encoding. This can be done using the *linear combination of unitaries lemma* as proven by [BCK15]. We consider a slightly more general version of their result, for which we first define State-preparation pairs:

Definition 2.8 : State-preparation pair *Let $b, m \in N$ be such that $2^b \geq m$. Let $y \in \mathbb{C}^m$, and $\beta \geq \|y\|_1$. The pair of unitaries (P_L, P_R) is called a (β, b, ε)*-state-preparation pair *for y if $P_L|0\rangle^{\otimes b} = \sum_{j=0}^{2^b-1} c_j|j\rangle$ and $P_R|0\rangle^{\otimes b} = \sum_{j=0}^{2^b-1} d_j|j\rangle$ such that $\sum_{j=0}^{m-1}\left|\beta\cdot\left(c_j^* d_j\right) - y_j\right| \leq \varepsilon$ and for all $j \in m,\dots,2^b-1$ we have $c_j^* d_j = 0$. A* symmetric *state-preparation pair also satisfies $c_j = d_j$ for all $j \in 0,\dots,m-1$.*

As we will show below, the naming of such a pair is not a coincident, a state-preparation pair can be used to prepare a state with amplitudes that are proportional to the y_j. Consider the unitary

$$\left(I \otimes P_L^\dagger\right) COPY \left(I \otimes P_R\right)$$

where $COPY$ copies a computational basis state in the second register to the first register. Now, if $\varepsilon = 0$ then applying this unitary to the all-zero state, and post-selecting on the second register being zero, gives

$$\begin{aligned}
&(I\otimes\langle 0|)\left(I\otimes P_L^\dagger\right)COPY\left(I\otimes P_R\right)(|0\rangle\otimes|0\rangle)\\
&\qquad=\left(I\otimes\sum_{i=0}^{2^b-1}c_i^*\langle i|\right)COPY\left(|0\rangle\otimes\sum_{j=0}^{2^b-1}d_j\big|j\big\rangle\right)\\
&\qquad=\left(I\otimes\sum_{i=0}^{2^b-1}c_i^*\langle i|\right)\left(\sum_{j=0}^{2^b-1}d_j\big|j\big\rangle\otimes\big|j\big\rangle\right)\\
&\qquad=\sum_{i=0}^{2^b-1}\sum_{j=0}^{2^b-1}c_i^*d_j\big|j\big\rangle\otimes\big\langle i\big|j\big\rangle\\
&\qquad=\sum_{j=0}^{2^b-1}c_j^*d_j\big|j\big\rangle\\
&\qquad=\sum_{j=0}^{2^b-1}y_j/\beta\big|j\big\rangle.
\end{aligned}$$

Hence this unitary is a block-encoding of a matrix that has the vector y/β in its first column.

Lemma 2.9 : Linear combination of block-encodings *Let $A=\sum_{j=0}^{m-1}y_jA_j$ be a w-qubit operator and $\varepsilon\in\mathbb{R}_{>0}$. Suppose that (P_L,P_R) is a $\left(\beta,b,\varepsilon_1\right)$-state-preparation pair for y,*

$$W=\sum_{j=0}^{m-1}U_j\otimes\big|j\big\rangle\big\langle j\big|+\left(I_{2^{w+a}}\otimes\left(I_{2^b}-\sum_{j=0}^{m-1}\big|j\big\rangle\big\langle j\big|\right)\right)$$

is an $w+a+b$ qubit unitary such that for all $j\in 0,\ldots,m$ we have that U_j is an $(1,a,\varepsilon_2)$-block-encoding of A_j. Then we can implement a $\left(\beta,a+b,\varepsilon_1+\beta\varepsilon_2\right)$-block-encoding of A, with a single use of W, P_R and $P_L^\dagger$.

Proof. With a calculation similar to the one for state-preparation pairs we can see that

$$\widetilde{W}=\left(I_{2^w}\otimes I_{2^a}\otimes P_L^\dagger\right)W\left(I_{2^w}\otimes I_{2^a}\otimes P_R\right)$$

is a $\left(\beta,a+b,\varepsilon_1+\beta\varepsilon_2\right)$-block-encoding of A. □

The linear combination of block-encodings lemma, combined with taking the powers of unitaries by repeatedly applying them, allows us to create a block-encoding of a polynomial applied to a matrix. This in turn allows us to implement approximations of smooth functions applied to matrices by (for example) implementing a truncated Taylor expansion of the function. We give an example of how this can be useful below.

Example: Hamiltonian simulation Possibly the most useful application of quantum computing is simulating quantum mechanics. Since quantum systems evolve via a unitary of the form $e^{\mathbf{i}H}$ for a Hermitian *Hamiltonian* H, we would like to simulate this type of dynamics. We will now give a rough sketch of how this could be done. Let us assume that we can apply a controlled version of a $(1,0,0)$-block-encoding of H, which is denoted by V, and let us assume that $\|H\| \leq 1/2$. Consider the function $f(x) = e^{\mathbf{i}x}$ from the reals to the complex unit circle. Then clearly $f(H)$ is the unitary corresponding to $e^{\mathbf{i}H}$. We can write $f(x)$ as a Taylor series

$$f(x) = e^{\mathbf{i}x} = \sum_{k=0}^{\infty} \frac{\mathbf{i}^k}{k!} x^k.$$

On the interval $[-1/2, 1/2]$ we can approximate f up to additive error ε by truncating the Taylor series after $p = \max\{2\log(1/\varepsilon), 2e\}$ terms:

$$\begin{aligned}\left| e^{\mathbf{i}x} - \sum_{k=0}^{p} \frac{\mathbf{i}^k}{k!} x^k \right| &= \left| \sum_{k=p+1}^{\infty} \frac{\mathbf{i}^k}{k!} x^k \right| \\ &\leq \sum_{k=p+1}^{\infty} \left| \frac{x^k}{k!} \right| \\ &\leq \sum_{k=p+1}^{\infty} \left(\frac{e}{2k} \right)^k \\ &\leq \sum_{k=p+1}^{\infty} \left(\frac{e}{2p+2} \right)^k \\ &= \left(\frac{e}{2p+2} \right)^{p+1} \frac{1}{1 - \frac{e}{2p+2}} \\ &\leq \varepsilon.\end{aligned}$$

Also note that $\sum_{k=0}^{p} \left| \frac{\mathbf{i}^k}{k!} \right| \leq e$.

Now let (P_L, P_R) be an $(e, \lfloor \log(p+1) \rfloor, 0)$-state-preparation pair for the $(p+1)$-dimensional vector y with coefficients $y_k = \frac{\mathbf{i}^k}{k!}$ (such a pair can efficiently be implemented). Now we use the linear combination of block-encodings lemma with $U_i = V^i$ to implement an $(e, 0, \varepsilon)$-block-encoding of $e^{\mathbf{i}H}$.

The following more general Hamiltonian simulation theorem is a corollary of the results of [LC16, Thm. 1]. For a detailed proof see the work of Chakraborty, Gilyén and Jeffery [CGJ18].

Theorem 2.10 : [LC16] *Suppose that U is an $(\alpha, a, \varepsilon/|2t|)$-block-encoding of the Hamiltonian H. Then we can implement an ε-precise Hamiltonian simulation unitary V which is an $(1, a+2, \varepsilon)$-block-encoding of $e^{\mathbf{i}tH}$, with $\mathcal{O}\left(|\alpha t| + \log(1/\varepsilon)\right)$ uses of controlled-U and its inverse and with $\mathcal{O}\left(a|\alpha t| + a\log(1/\varepsilon)\right)$ two-qubit gates.*

Finally we note that for real-valued polynomials that are bounded on $[-1,1]$ we can directly apply the following lemma in order to implement the polynomial applied to a block-encoding.

Lemma 2.11 : [GSLW19, Thm. 56] *Suppose that U is an (α, a, ε)-block-encoding of a Hermitian matrix A, $\delta > 0$ is a precision parameter, and $P \in \mathbb{R}[x]$ is a degree-d polynomial satisfying that*

(i) for all $x \in [-1,1]$: $|P(x)| \leq \frac{1}{2}$, or

(ii) for all $x \in [-1,1]$: $|P(x)| \leq 1$ and $|P(x)| = |P(-x)|$.

Then there is a quantum circuit $\tilde{U}$, which is an $(1, a+2, 4d\sqrt{\varepsilon/\alpha} + \delta)$-block-encoding of $P(A/\alpha)$, and which consists of d applications of U and $U^\dagger$ (and in item (i) a single application of controlled-U and controlled-$U^\dagger$), and $\mathcal{O}((a+1)d)$ other one- and two-qubit gates. Furthermore, we can compute the gates of this circuit on a classical computer in time $\mathcal{O}\big(\mathrm{poly}\big(d, \log(1/\delta)\big)\big)$.

2.4 | Basics of convex optimization

Optimization is a fundamental area in mathematics and computer science, with many real-world applications. One of the most successful continuous optimization paradigms is *convex* optimization. Below we give a short introduction to some basic concepts in convex optimization. We start with a look at general convex optimization problems. After this we consider two specific classes of convex optimization problems: linear programs and semidefinite programs. For a more in-depth look at convex optimization we recommend the book *Convex optimization* by S. Boyd and L. Vandenberghe [BV04].

2.4.1 | General convex optimization

Let us start by defining the k-dimensional *simplex* as the set

$$\Delta^k = \left\{ x \in \mathbb{R}^k : \|x\|_1 = 1, 0 \leq x_i \leq 1 \forall i \in [k] \right\}.$$

A *convex combination* of a set of points is a linear combination of those points where the vector of coefficients comes from the simplex.

A set $C \subseteq \mathbb{R}^n$ is called a *convex set* if every convex combination of points in the set also lies in the set, or equivalently, if for all points $x, y \in C$ and all $\lambda \in [0,1]$ the point $\lambda x + (1-\lambda) y$ also lies in C. Some examples of convex sets are:

- The empty set is convex, as is every singleton set, and so is the whole of $\mathbb{R}^n$.
- Affine subspaces are convex, as are half-spaces.

- The *convex hull* of a set of points, which consists of all convex combinations of the points, is convex.
- The set of positive semidefinite matrices is convex.
- The intersection of convex sets is convex.
- The Cartesian product of convex sets is again convex.

A function $f : C \to \mathbb{R}$ is called a *convex function* if the region above the graph of f (called the *epigraph*) is convex, or equivalently, if for all $x, y \in C$ we have

$$f\big(\lambda x + (1-\lambda)y\big) \leq \lambda f(x) + (1-\lambda) f(y).$$

A function f is called *concave* if $-f$ is convex. Some examples of convex functions are:

- Linear functions, which follows directly from the definitions.
- Norms, which follows from the triangle inequality.
- The max function, and also the *log-sum-exp* function $\log\big(\sum_i e^{x_i}\big)$ which is a smooth approximation of the max.
- The $\log(\det X)$ for matrices $X \in \mathbb{R}^{n\times n}$.
- The composition of two convex functions is again convex.

In general a convex optimization problem asks to minimize a convex function $f(x)$ over a convex set C (or maximize a concave function):

$$\inf_{x\in C} f(x).$$

Here f is called the *objective function* or simply the *objective*. We call the value of the infimum the *optimal value* of the problem, and an x that attains this value is called an *optimizer*. We often write x^* for such an optimizer if the infimum is attained.

An important characteristic of convex optimization problems is that *local minima* are also *global minima*. This means that optimization techniques may be somewhat greedy when improving their solutions, and that by improving the solution locally we will eventually get close to a global minimizer (if one exists). Of course this does not mean that such an algorithm will be efficient in finding a solution.

Instead of only considering f we may also consider its derivatives. Of particular interest is the first derivative, the *gradient*, since this tells us in which direction f decreases most. Since f does not need to have a derivative at every point of the domain we have the following more general notion of a *subgradient*.

Definition 2.12 : Subgradient *Let $C \subseteq \mathbb{R}^n$ be convex and let x be an element of the interior of C. For a convex function $f : C \to \mathbb{R}$ we denote by $\underline{\partial} f(x)$ the set of subgradients of f at x, i.e., those vectors g satisfying*

$$f(y) \geq f(x) + \langle g, y - x \rangle \text{ for all } y \in C.$$

Note that in the above definition $\underline{\partial} f(x) \neq \emptyset$ due to convexity and that if $\nabla f(x)$ exists, then $\underline{\partial} f(x) = \{\nabla f(x)\}$.

If $f : C \to \mathbb{R}$ is L-Lipschitz, then for any x in the interior of C and any $g \in \underline{\partial} f(x)$ we have $\|g\| \leq L$, as follows. Consider a $y \in C$ such that $y - x = \alpha g$ for some $\alpha > 0$. Then since g is a subgradient of f at x we have

$$\alpha \|g\|^2 = \langle g, y - x \rangle \leq f(y) - f(x) \leq L \|y - x\| = \alpha L \|g\|, \tag{2.2}$$

and therefore $\|g\| \leq L$.

An important property of convex sets is that any point outside of the set can be separated from the set by a *separating hyperplane*. Formally this means that for all $x \notin C$ there exists a $g \in \mathbb{R}^n$ such that $\|g\| = 1$ and

$$\langle y, g \rangle \leq \langle x, g \rangle \text{ for all } y \in C.$$

Note that this expression is very similar to the definition of a subgradient. In fact, a subgradient is a hyperplane that separates the epigraph of a function from a point very close to the epigraph.

Often the convex set C is described via a set of convex functions which should be smaller than some constant, these inequalities are called *constraints*:

$$C = \{x \in \mathbb{R}^n : h_j(x) \leq b_j \forall j \in [m]\}$$

for some $m \in \mathbb{N}$. By absorbing the b_j into the h_j we can assume without loss of generality that the right-hand sides are all zero. We can now write the optimization problem as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & h_1(x) \leq 0 \\ & \vdots \\ & h_m(x) \leq 0. \end{aligned}$$

For a point $x \notin C$ we say the constraint h_j is *violated* if $h_j(x) > 0$ and it is *satisfied* if $h_j(x) \leq 0$. If h_j is violated then for every $g \in \underline{\partial} h_j(x)$ the inequality $\langle g, y \rangle \leq h_j(x)$ describes a separating hyperplane for C and x. A point that satisfies all constraints is called a *feasible point*, a point that satisfies all constraints with strict inequality is called *strictly feasible*. An optimization problem that has a feasible point is called a *feasible problem*, and similarly for strictly feasible.

Every point x that is feasible certifies that the optimal value is less than or equal to $f(x)$. When optimizing we often would like to give lower bounds on the optimal value as well, since the difference between the upper and lower bound gives us a bound on how far we are from the optimal value. Consider the following function, called a *Lagrangian*:

$$\mathscr{L} : \mathbb{R}^n \times \mathbb{R}^m : (x, y) \mapsto f(x) + \sum_{j=1}^{m} y_j h_j(x).$$

The Lagrangian adds the constraints to the objective as a penalizing term, if $y_j > 0$ then to minimize the Lagrangian we would like to ensure that $h_j(x) \le 0$. In fact we can write down the *Lagrangian dual function* that minimizes over x for a fixed choice of y:

$$g(y) = \inf_{x \in \mathbb{R}^n} \mathscr{L}(x, y) = \inf_{x \in \mathbb{R}^n} f(x) + \sum_{j=1}^{m} y_j h_j(x).$$

Let x be a feasible point for the original problem and let $y \ge 0$, then

$$g(y) \le f(x)$$

because all $h_j(x) \le 0$. Since this is true for all feasible x, we in fact know that $g(y)$ lower bounds the optimal value of the original problem. The best lower bound this method can give us is then given by

$$\sup_{y \in \mathbb{R}^m_{\ge 0}} g(y).$$

Since it can be shown that $g(y)$ is a concave function, this problem is again a convex optimization problem. This problem is called the *dual problem* of our original problem, the original problem is called the *primal problem*. The fact that

$$\sup_{y \in \mathbb{R}^m_{\ge 0}} g(y) \le \inf_{x \in C} f(x)$$

is called *weak duality*. Sometimes the optimal values are actually equal, in which case we say that *strong duality* holds. It can be shown that the dual of the dual problem is the primal problem, which justifies the names.

2.4.2 | Linear programming

Arguably the simplest class of convex optimization problems is the class of *linear programming problems* (LPs). These are optimization problems where both f and all the h_j are linear functions.[3] Without loss of generality we assume LPs are of the form

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & \langle c, x \rangle \\ \text{s.t.} \quad & \langle a_j, x \rangle \le b_j \quad \text{for all } j \in [m] \\ & x \ge 0. \end{aligned} \tag{2.3}$$

[3] We cannot absorb the b_j into the h_j since then these would become *affine*.

For normalization purposes we assume that all entries of the a_j and of c are in $[-1,1]$.

Note that since linear functions are both convex and concave, we may assume without loss of generality that we want to maximize the objective. We can rewrite this problem in a more concise form by letting $A \in \mathbb{R}^{m\times n}$ be the matrix with rows equal to the a_j's in order to obtain the following primal and dual problems:

$$\begin{aligned} \max_{x\in\mathbb{R}^n} \quad & \langle c, x\rangle \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \qquad\qquad \begin{aligned} \min_{y\in\mathbb{R}^m} \quad & \langle b, y\rangle \\ \text{s.t.} \quad & A^T y \geq c \\ & y \geq 0. \end{aligned}$$

If both of these problems are feasible, then strong duality holds [BV04] and the optimal values are equal (and attained). We call a primal (respectively dual) LP *bounded* if there exists an upper (respectively lower) bound on the objective value of all feasible points. We call an LP *unbounded* if it is not bounded. If a primal LP is unbounded then by weak duality the dual LP cannot be feasible (and similarly the other way around).

Consider a point x that is infeasible for the LP. Since x violates at least one of the constraints, it can be separated from the feasible region of the LP by a subgradient of that constraint. If x violates the constraint $\langle a_j, x\rangle \leq b_j$ then clearly a_j is the gradient of this constraint. In fact, the feasible region of an *LP* is described by the intersection of affine half-spaces corresponding to the a_j constraints and the $x_i \geq 0$ constraints. Such a region is called a *polytope*.[4] A point v in a polytope is called a *vertex* if it cannot be written as a convex combination of other points in the polytope. A polytope that is contained in a finite region can alternatively be described as the convex hull of all the vertices of the polytope. If the feasible region of an LP is bounded, then the LP always attains its optimal value at a vertex.

Example: A scheduling problem Many natural problems can be written as an LP, below we give a toy example of how this can be done. Imagine a fictional PhD student who wants to plan his week so he gets all his work done without too much of a toll on his mental health. There might be four things he can do at any moment: work on his thesis, think about new research, do administrative work or play foosball. For every day of the week he needs to allocate a number of hours to these activities, denoted by variables T_i, R_i, A_i and F_i for $i \in [7]$. Clearly all these variables should be non-negative. In a day at most 20 hours can reasonably be spent awake, which introduces the constraints $T_i + R_i + A_i + F_i \leq 20$ for all $i \in [7]$.

He might need to do at least 3 hours of administrative work, leading to a constraint $\sum_{i\in[7]} A_i \geq 3$. He also needs to work on his thesis for at least 30 hours but knows

[4] In some parts of the literature the term *polyhedron* is used to describe such regions, with a *polytope* being a bounded polyhedron. We will use the term polytope for both bounded and unbounded regions.

that he is half as productive on Friday, Saturday or Sunday, leading to the constraint $\sum_{i\in[4]} T_i + \frac{1}{2}\sum_{i\in\{5,6,7\}} T_i \geq 30$. A research meeting on Wednesday implies that $R_3 \geq 2$. In the weekends all colleagues are at home and hence foosball is not an option: $F_6 = 0$ and $F_7 = 0$. For all other days we can play at most an hour of foosball: $F_i \leq 1$ for $i \in [5]$.

Next we introduce variables M_i for the misery on day i. We assume that all misery will be non-negative. For the weekdays any time after 8 hours of work counts towards the misery for that day. However, since research is interesting it only counts for half towards the total, and foosball counts for half in the other direction. This gives the constraints $T_i + A_i + \frac{1}{2}R_i - \frac{1}{2}F_i \leq 8 + M_i$ for $i \in [5]$. In the weekend any time is overtime (and foosball is not an option) and hence the constraints are $T_i + A_i + \frac{1}{2}R_i \leq M_i$ for $i \in \{6,7\}$. Finally, if less than five hours of research is done then this is demotivating and adds some misery M_0, for each hour that is not done this misery increases by 3. This is modelled by the constraint $\sum_{i\in[7]} R_i + \frac{1}{3}M_0 \geq 5$.

Our objective is to minimize the sum of the misery. This leads to the following LP:

$$
\begin{aligned}
\min\quad & M_0 + \sum_{i\in[7]} M_i \\
\text{s.t.}\quad & T_i + R_i + A_i + F_i \leq 20 && \text{for all } i \in [7] \\
& \sum_{i\in[7]} A_i \geq 3 \\
& \sum_{i\in[4]} T_i + \frac{1}{2}\sum_{i\in\{5,6,7\}} T_i \geq 30 \\
& R_3 \geq 2 \\
& F_i \leq 1 && \text{for all } i \in [5] \\
& T_i + A_i + \frac{1}{2}R_i - \frac{1}{2}F_i \leq 8 + M_i && \text{for all } i \in [5] \\
& T_i + A_i + \frac{1}{2}R_i \leq M_i && \text{for all } i \in \{6,7\} \\
& \sum_{i\in[7]} R_i + \frac{1}{3}M_0 \geq 5 \\
& M_0, M_i, T_i, R_i, A_i, F_i \geq 0 && \text{for all } i \in [7].
\end{aligned}
$$

2.4.3 | Semidefinite programming

In the last decades, particularly since the work of Grötschel, Lovász, and Schrijver [GLS88], *semidefinite programs* (SDPs) have become an important tool for designing efficient optimization and approximation algorithms. SDPs generalize LPs, but are still convex and efficiently solvable. Thanks to their stronger expressive power, SDPs can sometimes give better approximation algorithms than LPs. Examples include approximation of NP-hard problems like MAXCUT [GW95] and polynomial optimization through the Sum-Of-Squares hierarchy [Las01; Par00]. SDPs have also found applications in quantum information theory. Examples include POVM design [Eld03] and finding the winning probability of non-local games [Tsi87; CHTW04].

In an SDP we optimize over a matrix X instead of a vector x. The objective and most of the constraints are still linear in the entries of X, in particular they are of the form $\mathrm{Tr}(MX)$ for a matrix M. However, the positivity constraint $x \geq 0$ is replaced by a positive semidefinite constraint $X \succeq 0$. An SDP can be written as

$$\begin{aligned} \max_{X\in\mathbb{R}^{n\times n}} \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}\left(A_j X\right) \leq b_j \quad \text{for all } j \in [m] \\ & X \succeq 0. \end{aligned} \tag{2.4}$$

For normalization purposes we may assume $\|C\|, \left\|A_j\right\| \leq 1$. LPs correspond to the case where all matrices are diagonal.

As for LPs we can write down the dual form of this SDP:

$$\begin{aligned} \min \quad & b^T y \\ \text{s.t.} \quad & \sum_{j=1}^{m} y_j A_j - C \succeq 0. \\ & y \geq 0. \end{aligned} \tag{2.5}$$

If the primal of the SDP is known to be bounded and strictly feasible, then we will assume an upper bound is known on the trace of a primal optimizer. In particular we will assume that $\mathrm{Tr}(X) \leq R$ is the first constraint in the primal. This ensures that the primal optimum is attained and we write R^* for minimal trace of a primal optimizer. The trace constraint in the primal will also ensure that the dual is strictly feasible. If we also know that the dual optimum is attained then we assume an upper bound on the ℓ_1-norm of the dual optimizer is known. In particular we assume that we know a r such that there exists a dual optimizer y with $\left\|y\right\|_1 \leq r$. We further write r^* for the smallest such r. For SDPs strong duality does not hold in general. However, in the situation described above the primal and dual optimal values are equal.

Example: MAXCUT and the Goemans-Williamson SDP A famous example of how semidefinite programming can be used is the algorithm for approximating the size of a maximum cut in a graph $G = ([n], E)$ by Goemans and Williamson [GW95]. The maximum cut in a graph is the maximum, over all subsets S of vertices, of the number of edges between S and its complement $\bar{S}$. Computing $\mathrm{MAXCUT}(G)$ exactly is NP-hard. It corresponds to the following integer program

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{\{i,j\}\in E} \left(1 - v_i v_j\right) \\ \text{s.t.} \quad & v_j \in \{+1, -1\} \quad \text{for all } j \in [n], \end{aligned}$$

using the fact that $\left(1 - v_i v_j\right)/2 = 1$ if v_i and v_j are different signs, and $\left(1 - v_i v_j\right)/2 = 0$ if they are the same. We can relax this integer program by replacing the signs v_j

by unit vectors, and replacing the product $v_i v_j$ in the objective function by the dot product $v_i^T v_j$. We can implicitly optimize over such vectors (of unspecified dimension) by explicitly optimizing over an $n \times n$ psd matrix X whose diagonal entries are 1. This X is the Gram matrix of the vectors $v_1, \dots, v_n$, so $X_{ij} = v_i^T v_j$. The resulting SDP is

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{\{i,j\} \in E} \left(1 - X_{ij}\right) \\ \text{s.t.} \quad & \operatorname{Tr}\left(E_{jj} X\right) = 1 \quad \text{for all } j \in [n] \\ & X \succeq 0. \end{aligned}$$

This SDP is a relaxation of a maximization problem, so it may overshoot the correct value, but Goemans and Williamson showed that an optimal solution to the SDP can be rounded to a cut in G whose size is within a factor ≈ 0.878 of $\mathrm{MAXCUT}(G)$ (which is optimal under the Unique Games Conjecture [KKMO04]). This SDP can be massaged into the form of (2.4) by replacing the equality $\operatorname{Tr}\left(E_{jj} X\right) = 1$ by inequality $\operatorname{Tr}\left(E_{jj} X\right) \leq 1$ (so $m = n$) and letting C be a properly normalized version of the Laplacian of G.

Part I

Quantum algorithms

Chapter 3

Convex optimization using quantum oracles

We study to what extent quantum algorithms can speed up solving convex optimization problems. Following the classical literature we assume access to a convex set via various oracles, and we examine the efficiency of reductions between the different oracles. In particular, we show how a separation oracle can be implemented using $\widetilde{\mathcal{O}}(1)$ quantum queries to a membership oracle, which is an exponential quantum speedup over the $\Omega(n)$ membership queries that are needed classically. We show that a quantum computer can very efficiently compute an approximate subgradient of a convex Lipschitz function. Combining this with a simplification of recent classical work of Lee, Sidford, and Vempala gives our efficient separation oracle. This in turn implies, via a known algorithm, that $\widetilde{\mathcal{O}}(n)$ quantum queries to a membership oracle suffice to implement an optimization oracle (the best known classical upper bound on the number of membership queries is quadratic.

This chapter is based on the paper *Convex optimization using quantum oracles* by J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf [AGGW18].

3.1 | Introduction

The generic problem in convex optimization is minimizing a convex function $f : K \to \mathbb{R} \cup \{\infty\}$, where $K \subseteq \mathbb{R}^n$ is a convex set. We consider the setting where an interior point $x_0 \in \text{int}(K)$ is given and radii $r, R > 0$ are known such that $B(x_0, r) \subseteq K \subseteq B(x_0, R)$.

It is well-known that if the convex function is bounded on K, then we can equivalently consider the problem of minimizing a *linear* function over a different convex set $K' \subseteq \mathbb{R}^{n+1}$, namely minimizing the linear function $(x, \mu) \mapsto \mu$ over the epigraph $K' = \{(x, \mu) : x \in K, f(x) \geq \mu\}$ of f. Accessing K' is easy given access to K and f, and the parameters involved will be similar. Conversely, for any linear optimization problem over an arbitrary convex set K, there is an equivalent optimization problem over a fixed convex set (say, the ball), with a bounded convex objective function f that can be evaluated easily given access to K. From now on we therefore focus on optimizing a linear function over a convex set.

In this chapter we consider the setting where access to the convex set is given only in a black-box manner, through an oracle. The five basic problems (oracles) in convex optimization identified by Grötschel, Lovász, and Schrijver [GLS88] are: membership, separation, optimization, violation, and validity (see Section 3.2 for the definitions). They showed that all five basic problems are polynomial-time equivalent. That is, given an oracle O for one of these problems, one can implement an oracle for any of the other problems using a polynomial number of calls to O and polynomially many other elementary operations. Subsequent work made these polynomial-time reductions more efficient, reducing the degree of the polynomials. Recently Lee et al. [LSV18] showed that with $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^2\right)$ classical calls to a membership oracle (and $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^3\right)$ other elementary arithmetic operations) one can solve an optimization problem. They did so by showing that $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ calls to a membership oracle suffice to do separation, and then composing this with the known fact [LSW15] (see also [LSV18, Thm. 15]) that $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ calls to a separation oracle suffice for optimization.

The main result of this chapter (Section 3.4) shows that on a quantum computer, $\widetilde{\mathcal{O}}_{\frac{Rn}{r\varepsilon}}(1)$ calls to a membership oracle suffice to implement a separation oracle, and hence (by the known classical reduction from optimization to separation) $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ calls to a membership oracle suffice for optimization.[1] Lee et al. [LSV18] use a geometric idea to reduce separation to finding an approximate subgradient of a convex Lipschitz

[1] Although not stated explicitly in our results, we also use $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^3\right)$ additional operations for optimization using membership, like [LSV18]. This is because our quantum algorithm for separation uses only $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ gates in addition to the $\widetilde{\mathcal{O}}_{\frac{Rn}{r\varepsilon}}(1)$ membership queries, and we use the same reduction from optimization to separation as [LSV18]. If queries themselves have significant time complexity, then our algorithm does lead to a speedup in time complexity over the best known classical algorithm. For example, if each membership query (with the required precision) takes time $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^2\right)$ to implement, then our quantum algorithm for optimization has time complexity $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^3\right)$, while the classical algorithm will use time $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^4\right)$ because it uses $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}\left(n^2\right)$ membership queries.

function. They then show that $\widetilde{\mathcal{O}}_{\frac{LR}{r\varepsilon}}(n)$ evaluations of a convex L-Lipschitz function suffice to get an approximate subgradient.

We use the same geometric idea, but we provide a simpler way to classically compute an approximate subgradient of a convex Lipschitz function (Section 3.3.1). Besides being simpler, the main advantage of our algorithm is that it is suitable for a quantum speedup using known quantum algorithms (Jordan's algorithm) for computing approximate (sub)gradients [Jor05; GAW19] as we show in Section 3.3.2. In Section 3.4 we use the new algorithms applied to the geometric idea from [LSV18] to construct separation oracles.

In Section 3.5 we briefly mention how to relate the discussed reductions to other reductions between the oracles, using a convex polarity argument. As we show, in the setting where we are given an interior point, the relation between membership and separation is analogous to the relation between validity and optimization. In particular, our better quantum algorithm for separation using membership queries implies that on a quantum computer $\widetilde{\mathcal{O}}_{\frac{Rn}{r\varepsilon}}(1)$ queries to a validity oracle suffice to implement an optimization oracle. That is, on a quantum computer, finding the optimal value is equivalent to finding an optimizer.

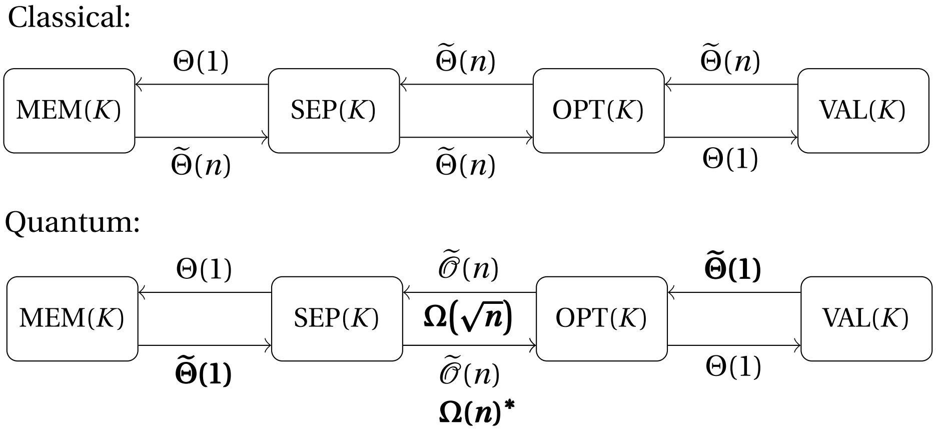


Figure 3.1: The top and bottom diagram illustrate the relations between the basic (weak) oracles for respectively classical and quantum queries, with boldface entries marking our new results. All upper and lower bounds are for the setting where we know an interior point of K, except the $*$-marked $\Omega(n)$ lower bound on the number of separation queries needed for optimization, which stems from a problem where no such point is known. Notice the central symmetry of the diagrams, which is a consequence of polarity. The lower bounds will be proven in Chapter 7. Here the notation $\widetilde{\mathcal{O}}(\cdot)$ is used to hide polylogarithmic factors in n, r, R, ε. The (change in) accuracy of the oracles is ignored here for simplicity

Related independent work In independent and simultaneous work, Chakrabarti, Childs, Li, and Wu [CCLW18] discovered a similar upper bound as ours: combining the recent classical work of Lee et al. [LSV18] with a quantum algorithm for computing gradients, they show how to implement an optimization oracle via $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ quantum queries to a membership oracle and to an oracle for the objective function. Their proof stays quite close to [LSV18] while ours first simplifies some of the technical lemmas of [LSV18], giving us a slightly simpler presentation and a better error-dependence of the resulting algorithm.

3.2 | Oracles for convex sets

The five basic oracles for a convex set K that we consider are as follows (in contrast with the original [GLS88], we allow some error probability ρ in these oracles as in [LSV18]). Throughout we will assume that real vectors are represented with $\mathrm{polylog}(nR/(r\varepsilon))$ bits of precision per coordinate. In particular, we will assume that both the input and output of the following oracles are represented in such a way.

Definition 3.1 : Membership oracle $\mathrm{MEM}_{\varepsilon,\rho}(K)$ *Queried with a vector $y \in \mathbb{R}^n$, the oracle, with success probability $\geq 1-\rho$, asserts one of the following*

- *$y \in B(K,\varepsilon)$, or*
- *$y \notin B(K,-\varepsilon)$.*

Definition 3.2 : Separation oracle $\mathrm{SEP}_{\varepsilon,\rho}(K)$ *Queried with a vector $y \in \mathbb{R}^n$, the oracle, with success probability at least $\geq 1-\rho$, asserts one of the following*

- *$y \in B(K,\varepsilon)$, or*
- *$y \notin B(K,-\varepsilon)$,*

and in the second case it returns a unit vector $g \in \mathbb{R}^n$ such that $\langle g,x\rangle \leq \langle g,y\rangle + \varepsilon$ for all $x \in B(K,-\varepsilon)$.

Definition 3.3 : Optimization oracle $\mathrm{OPT}_{\varepsilon,\rho}(K)$ *Queried with a unit vector $c \in \mathbb{R}^n$, the oracle, with success probability $\geq 1-\rho$, does one of the following:*

- *it returns a vector $y \in \mathbb{R}^n$ such that $y \in B(K,\varepsilon)$ and $\langle c,x\rangle \leq \left\langle c,y\right\rangle + \varepsilon$ for all $x \in B(K,-\varepsilon)$,*
- *or it asserts that $B(K,-\varepsilon)$ is empty.*

Note that the above optimization oracle corresponds to *maximizing* a linear function over a convex set; we could equally well state it for minimization.

Definition 3.4 : Violation oracle $\mathrm{VIOL}_{\varepsilon,\rho}(K)$ *Queried with a unit vector $c \in \mathbb{R}^n$ and a real number γ, the oracle, with success probability $\geq 1-\rho$, does one of the following:*

- *it asserts that $\langle c,x\rangle \leq \gamma + \varepsilon$ for all $x \in B(K,-\varepsilon)$,*

- *or it finds a vector* $y \in B(K,\varepsilon)$ *such that* $\langle c, y\rangle \geq \gamma - \varepsilon$.

Definition 3.5 : Validity oracle $\mathrm{VAL}_{\varepsilon,\rho}(K)$ *Queried with a unit vector* $c \in \mathbb{R}^n$ *and a real number* γ*, the oracle, with success probability* $\geq 1-\rho$*, does one of the following:*

- *it asserts that* $\langle c, x\rangle \leq \gamma + \varepsilon$ *for all* $x \in B(K,-\varepsilon)$*,*
- *or it asserts that* $\langle c, y\rangle \geq \gamma - \varepsilon$ *for some* $y \in B(K,\varepsilon)$.

If in the above definitions both ε and ρ are equal to 0, then we call the oracle *strong*. If either is non-zero then we sometimes call it *weak*.

When discussing quantum oracles we still assume that the inputs and outputs of the oracles are represented with $\mathrm{polylog}(nR/(r\varepsilon))$ bits of precision per coordinate. Since the oracles are randomized and might themselves come from quantum algorithms, it is natural to assume they are relational quantum oracles (see Definition 2.1). Throughout this chapter we will work with deterministic quantum oracles for the sake of readability. However, the results presented are also valid for relational oracles by using a version of the quantum gradient algorithm for relation oracles [AGGW18, Cor. 29]. In terms of applications, we want to point out that if the membership oracle used in Section 3.4 comes from a deterministic algorithm, then we get a standard quantum oracle. Only when the membership oracle itself is relational (for example, when it is itself computed by a bounded-error quantum algorithm) do we need the more general setting of [AGGW18, Cor. 29].

When we discuss membership queries, we will always assume that we are given a small ball which lies inside the convex set. It is easy to see that without such a small ball one cannot obtain an optimization oracle using only $\mathrm{poly}(n)$ classical queries to a membership oracle (see, e.g., [GLS88, Sec. 4.1] or the example below). As the following example shows, the same holds for quantum queries. We will use a reduction from a version of the well-studied *search* problem:

Given $z \in \{0,1\}^N$ *such that* $|z| = 1$*, find* $b \in [N]$ *such that* $z_b = 1$.

It is not hard to see that if the access to z is given via classical queries $i \mapsto z_i$, then $\Omega(N)$ queries are needed. It is well known that if we allow quantum queries then $\Omega(\sqrt{N})$ queries are needed (see Chapter 6). Now let $N = 2^n$ and consider an input $z \in \{0,1\}^N$ to the search problem (see also Section 2.3.1). Let $b \in \{0,1\}^n$ be the index such that $z_b = 1$. Consider maximizing the linear function $x \mapsto \sum_{i=1}^n x_i$ over the set $K_z = \prod_{i=1}^n [b_i - 1/2, b_i]$. Clearly the optimal solution to this convex optimization problem, even with a small constant additive error in the answer, gives the solution to the search problem. However, a membership query is essentially equivalent to querying a bit of z and therefore $\Omega(\sqrt{N}) = \Omega(2^{n/2})$ quantum queries to the membership oracle are needed to solve this optimization problem.

3.3 | Subgradient approximation for convex Lipschitz functions

Here we show how to compute an approximate subgradient (at 0) of a convex Lipschitz function. That is, given a convex set C such that $0 \in \mathrm{int}(C)$ and a convex function $f : C \to \mathbb{R}$, we show how to compute a vector $\tilde{g} \in \mathbb{R}^n$ such that

$$f(y) \geq f(0) + \langle \tilde{g}, y \rangle - a\|y\| - b \text{ for all } y \in C$$

for some real numbers $a, b > 0$ that will be defined later (see Lemma 3.10 and Lemma 3.13). The idea of the classical algorithm given in the next section is to pick a point $z \in B_\infty(0, r_1)$ uniformly at random and use the finite difference $\nabla^{(r_2)} f(z)$ (defined below) as an approximate subgradient of f at 0; the radii r_1 and r_2 need to be chosen small to make the approximation good. This results in a slightly simplified version of the algorithm of Lee et al. [LSV18]. In Section 3.3.2 we show how to improve on this classical algorithm on a quantum computer.

3.3.1 | Classical approach

In the discussion that follows we will use the following approximation of the gradient.

Definition 3.6 : Finite-difference gradient approximation *For a function $f : C \to \mathbb{R}$, a real $r > 0$, and a point $x \in \mathbb{R}^n$ such that $B_1(x, r) \subseteq C$, and $i \in [n]$, we define*

$$\nabla_i^{(r)} f(x) := \frac{f(x + re_i) - f(x - re_i)}{2r}.$$

Similarly we define

$$\nabla^{(r)} f(x) := \left(\nabla_1^{(r)} f(x), \nabla_2^{(r)} f(x), \ldots, \nabla_n^{(r)} f(x)\right).$$

We will also consider a similar approximation of the Laplacian (the trace of the Hessian) of a function.

Definition 3.7 : Finite-difference Laplace approximation *For a function $f : C \to \mathbb{R}$, a real $r > 0$, and a point $x \in \mathbb{R}^n$ such that $B_1(x, r) \subseteq C$, and $i \in [n]$, we define*

$$\Delta_i^{(r)} f(x) := \frac{f(x + re_i) - 2f(x) + f(x - re_i)}{r^2}.$$

Similarly

$$\Delta^{(r)} f(x) := \sum_{i=1}^{n} \Delta_i^{(r)} f(x).$$

Note that for a convex function we have $\Delta_i^{(r)} f(x) \geq 0$ for all x such that $B_1(x,r) \subseteq C$.

The next two lemmas will be needed in the proof of the main result of this section, Lemma 3.10. In Lemma 3.8 we give an upper bound on the deviation $\left\|g - \nabla^{(r_2)} f(z)\right\|_1$ of a finite difference gradient approximation $\nabla^{(r_2)} f(z)$ from an actual subgradient g at the point z, in terms of the finite difference Laplace approximation $\Delta^{(r_2)} f(z)$. Then, in Lemma 3.9 we show that in expectation over the choice of z, the finite difference Laplace approximation is small. Together with Markov's inequality this gives us good control over the quality of a finite difference gradient approximation.

Lemma 3.8 *If $r_2 > 0$, $z \in \mathbb{R}^n$, and $f : B_1(z,r_2) \to \mathbb{R}$ is convex, then*

$$\sup_{g \in \underline{\partial} f(z)} \left\|g - \nabla^{(r_2)} f(z)\right\|_1 \leq \frac{r_2 \Delta^{(r_2)} f(z)}{2}.$$

Proof. Fix a $g \in \underline{\partial} f(z)$. For every $i \in [n]$, we have

$$f(z + r_2 e_i) \geq f(z) + \langle g, r_2 e_i \rangle = f(z) + r_2 g_i,$$

and, similarly,

$$f(z - r_2 e_i) \geq f(z) - r_2 g_i.$$

Rearranging these two equations gives

$$\underbrace{\frac{f(z) - f(z - r_2 e_i)}{r_2}}_{:=A} \leq g_i \leq \underbrace{\frac{f(z + r_2 e_i) - f(z)}{r_2}}_{:=B}.$$

Note that $\left|g_i - \frac{A+B}{2}\right| \leq \frac{B-A}{2}$ for any three real numbers $A \leq g_i \leq B$. Moreover, $\frac{A+B}{2} = \nabla_i^{(r_2)} f(z)$ and $B - A = r_2 \Delta_i^{(r_2)} f(z)$, thus

$$\left|g_i - \nabla_i^{(r_2)} f(z)\right| \leq \frac{r_2 \Delta_i^{(r_2)} f(z)}{2}.$$

Now we can finish the proof by summing this inequality over all $i \in [n]$. □

Lemma 3.9 *If $0 < r_2 \leq r_1$, and $f : B_\infty(x, r_1 + r_2) \to \mathbb{R}$ is convex and L-Lipschitz, then*

$$\mathop{\mathbb{E}}_{z \in B_\infty(x,r_1)} \Delta^{(r_2)} f(z) \leq \frac{nL}{r_1}.$$

Proof. Below we show that $\mathop{\mathbb{E}}_{z \in B_\infty(x,r_1)} \Delta_i^{(r_2)} f(z) \leq \frac{L}{r_1}$ for all $i \in [n]$, summing over i then proves the lemma. Let $h_i(z) := f(z - r_2 e_i) - f(z)$. We have that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{z\in B_\infty(x,r_1)} \Delta_i^{(r_2)} f(z)\\
&= \frac{1}{(2r_1)^n}\int_{z\in B_\infty(x,r_1)} \frac{f(z+r_2e_i)-2f(z)+f(z-r_2e_i)}{r_2^2}dz\\
&= \frac{1}{(2r_1)^n}\int_{\substack{z_j\in[x_j-r_1,x_j+r_1],\\ j\in[n],j\neq i}}\int_{z_i\in[x_i-r_1,x_i+r_1]} \frac{f(z-r_2e_i)-2f(z)+f(z+r_2e_i)}{r_2^2}dz\\
&= \frac{1}{(2r_1)^n}\int_{\substack{z_j\in[x_j-r_1,x_j+r_1],\\ j\in[n],j\neq i}}\Bigg(\int_{z_i\in[x_i-r_1,x_i+r_1]} \frac{h_i(z)}{r_2^2}dz_i\\
&\qquad\qquad -\int_{z_i\in[x_i-r_1,x_i+r_1]} \frac{h_i(z+r_2e_i)}{r_2^2}dz_i\Bigg)\prod_{\substack{j\in[n],\\ j\neq i}} dz_j\\
&= \frac{1}{(2r_1)^n}\int_{\substack{z_j\in[x_j-r_1,x_j+r_1],\\ j\in[n],j\neq i}}\Bigg(\int_{z_i\in[x_i-r_1,x_i+r_1]} \frac{h_i(z)}{r_2^2}dz_i\\
&\qquad\qquad -\int_{z_i\in[x_i-r_1+r_2,x_i+r_1+r_2]} \frac{h_i(z)}{r_2^2}dz_i\Bigg)\prod_{\substack{j\in[n],\\ j\neq i}} dz_j\\
&= \frac{1}{(2r_1)^n}\int_{\substack{z_j\in[x_j-r_1,x_j+r_1],\\ j\in[n],j\neq i}}\Bigg(\int_{z_i\in[x_i-r_1,x_i-r_1+r_2]} \frac{h_i(z)}{r_2^2}dz_i\\
&\qquad\qquad -\int_{z_i\in[x_i+r_1,x_i+r_1+r_2]} \frac{h_i(z)}{r_2^2}dz_i\Bigg)\prod_{\substack{j\in[n],\\ j\neq i}} dz_j\\
&\le \frac{1}{(2r_1)^n}\int_{\substack{z_j\in[x_j-r_1,x_j+r_1],\\ j\in[n],j\neq i}} 2L\prod_{\substack{j\in[n],\\ j\neq i}} dz_j\\
&= \frac{L}{r_1}.
\end{aligned}$$

Where the last inequality follows from multiplying the upper bound r_2L on $|h_i(z)|$ with the length r_2 of the integration intervals. □

Note that the above lemma is stated and proved for continuous random variables, but the same proof holds if we have a uniform hypergrid over the same hypercube, providing a discrete version of the above result. In the discrete case, in order to get the same cancellations we need to assume that both r_1 and r_2 are integer multiples of the grid spacing.

We are now ready to prove the main result of this section. Informally, the next lemma proves that an approximate subgradient of a convex Lipschitz function f at 0 can be obtained by an algorithm that outputs $\nabla^{(r_2)}\tilde{f}(z)$ for a random z close enough to 0, where $\tilde{f}$ is an approximate version of f. In other words, this lemma gives us a classical algorithm to compute an approximate subgradient of f using $2n$ classical queries to an approximate version of f.

Lemma 3.10 *Let $r_1 > 0$, $L > 0$, $\rho \in (0,1/3]$, $\delta \in (0, r_1\sqrt{n}L/\rho]$, then $r_2 := \sqrt{\frac{\delta r_1 \rho}{\sqrt{n}L}} \le r_1$. Suppose $f : C \to \mathbb{R}$ is a convex function that is L-Lipschitz on $B_\infty(0,2r_1)$, and $\tilde{f} : B_\infty(0,2r_1) \to \mathbb{R}$ is such that $\|\tilde{f} - f\|_\infty \le \delta$. Then for a uniformly random $z \in B_\infty(0,r_1)$, with probability at least $1-\rho$,*

$$f(y) \ge f(0) + \left\langle \nabla^{(r_2)}\tilde{f}(z), y \right\rangle - \frac{3n^{\frac{3}{4}}}{2}\sqrt{\frac{\delta L}{\rho r_1}}\|y\| - 2L\sqrt{n}r_1 \qquad \textit{for all } y \in C.$$

Proof. Let $z \in B_\infty(0,r_1)$ and $g \in \underline{\partial} f(z)$. Recall $\|g\| \le L$ by Equation (2.2). Then for all $y \in C$

$$\begin{aligned} f(y) &\ge f(z) + \langle g, y-z\rangle \\ &= f(z) + \langle g, y-z\rangle + \left(\left\langle \nabla^{(r_2)} f(z), y\right\rangle - \left\langle \nabla^{(r_2)} f(z), y\right\rangle\right) + \left(f(0) - f(0)\right) \\ &= f(0) + \left\langle \nabla^{(r_2)} f(z), y\right\rangle + \langle g - \nabla^{(r_2)} f(z), y\rangle + (f(z) - f(0)) + \langle g, -z\rangle \\ &\ge f(0) + \left\langle \nabla^{(r_2)} f(z), y\right\rangle - \left\|g - \nabla^{(r_2)} f(z)\right\|_1 \|y\|_\infty - L\|z\| - \|g\|\|z\| \\ &\ge f(0) + \left\langle \nabla^{(r_2)} f(z), y\right\rangle - \left\|g - \nabla^{(r_2)} f(z)\right\|_1 \|y\|_\infty - L\sqrt{n}r_1 - L\sqrt{n}r_1 \\ &\ge f(0) + \left\langle \nabla^{(r_2)} \tilde{f}(z), y\right\rangle - \frac{\delta\sqrt{n}}{r_2}\|y\| - \left\|g - \nabla^{(r_2)} f(z)\right\|_1 \|y\|_\infty - 2L\sqrt{n}r_1. \end{aligned}$$

Note that in the last line we switched from f to $\tilde{f}$, using that $\nabla^{(r_2)} f(z)$ and $\nabla^{(r_2)}\tilde{f}(z)$ differ by at most δ/r_2 in each coordinate. Our choice of r_2 gives $\frac{\delta\sqrt{n}}{r_2} = n^{\frac{3}{4}}\sqrt{\frac{\delta L}{\rho r_1}}$ and by Lemma 3.8 and Lemma 3.9 we have

$$\mathop{\mathbb{E}}_{z \in B_\infty(x,r_1)} \left\|g - \nabla^{(r_2)} f(z)\right\|_1 \le \frac{nLr_2}{2r_1} = \frac{n^{\frac{3}{4}}}{2}\sqrt{\frac{\delta L \rho}{r_1}}.$$

By Markov's inequality we get that $\left\|g - \nabla^{(r_2)} f(z)\right\|_1 \le \frac{n^{\frac{3}{4}}}{2}\sqrt{\frac{\delta L}{\rho r_1}}$ with probability $\ge 1-\rho$ over the choice of z. Plugging this bound on $\left\|g - \nabla^{(r_2)} f(z)\right\|_1$ into the above lower bound on $f(y)$, and using that $\|y\|_\infty \le \|y\|$, concludes the proof of the lemma. □

3.3.2 | Quantum improvements

In this section we show how to improve subgradient computation of convex functions via Jordan's quantum algorithm for gradient computation [Jor05]. We use the formulation given by Gilyén et al. [GAW19, Lem. 21], as introduced in Section 2.3.3.

In order to apply Lemma 2.6 the function needs to be sufficiently close to linear on a small region. Fortunately, convex Lipschitz functions can be very well approximated by linear functions over most small-enough regions. Similarly to the classical case

(Lemma 3.10) we make this claim quantitative using Lemma 3.9. In order to apply the more efficient quantum gradient computation of Lemma 2.6 we also need the following two lemmas to ensure that Equation (2.1) holds.

Lemma 3.11 *Let $S \subseteq \mathbb{R}^n$ be such that $S = -S$, and let* $\mathrm{conv}(S)$ *denote the convex hull of S. If $f : \mathrm{conv}(S) \to \mathbb{R}$ is a convex function, $f(0) = 0$, and $\left|f(s)\right| \le \delta$ for all $s \in S$, then*

$$\left|f(s')\right| \le \delta \textit{ for all } s' \in \mathrm{conv}(S).$$

Proof. Since f is convex and $f(s) \le \delta$ for all $s \in S$ we immediately get that $f(s') \le \delta$ for all $s' \in \mathrm{conv}(S)$. Because $f(0) = 0$ and $S = -S$, due to convexity we get that $f(s') \ge -f(-s') \ge -\delta$. □

Lemma 3.12 *If $r_2 > 0$, $z \in \mathbb{R}^n$ and $f : B_1(z, r_2) \to \mathbb{R}$ is convex, then*

$$\sup_{y \in B_1(0,r_2)} \left|f(z+y) - f(z) - \left\langle y, \nabla^{(r_2)} f(z)\right\rangle\right| \le \frac{r_2^2 \Delta^{(r_2)} f(z)}{2}.$$

Proof. Let $d(y) := f(z+y) - f(z) - \left\langle y, \nabla^{(r_2)} f(z)\right\rangle$ be the difference between $f(z+y)$ and its linear approximator. Let $S := \{\pm r_2 e_i : i \in [n]\}$. It is easy to see that $d(0) = 0$, $S = -S$, and $\mathrm{conv}(S) = B_1(0, r_2)$. Also, for all $s \in S$ we have $|d(s)| \le r_2^2 \Delta^{(r_2)} f(z)/2$:

$$\begin{aligned}
d(\pm r_2 e_i) &= f(z \pm r_2 e_i) - f(z) - \left\langle \pm r_2 e_i, \nabla^{(r_2)} f(z)\right\rangle \\
&= f(z \pm r_2 e_i) - f(z) \mp r_2 \nabla_i^{(r_2)} f(z) \\
&= f(z \pm r_2 e_i) - f(z) \mp \frac{f(z + r_2 e_i) - f(z - r_2 e_i)}{2} \\
&= \frac{f(z + r_2 e_i) - 2f(z) + f(z - r_2 e_i)}{2} \\
&= r_2^2 \Delta_i^{(r_2)} f(z)/2 \quad \le r_2^2 \Delta^{(r_2)} f(z)/2.
\end{aligned}$$

Therefore Lemma 3.11 implies that $\sup_{y \in B_1(0,r_2)} \left|d(y)\right| \le r_2^2 \Delta^{(r_2)} f(z)/2$. □

Using the results above we are now ready to prove the main result of this section, the quantum analogue of Lemma 3.10.

Lemma 3.13 *Let $r_1 > 0$, $L > 0$, $\rho \in (0, 1/3]$, and suppose $\delta \in (0, r_1 nL/\rho]$. Suppose $f : C \to \mathbb{R}$ is a convex function that is L-Lipschitz on $B_\infty(0, 2r_1)$, and we have quantum query access[2] to $\tilde{f}$, which is a δ-approximate version of f, via a unitary U over a (fine-enough) hypergrid of $B_\infty(0, 2r_1)$. Then we can compute a $\tilde{g} \in \mathbb{R}^n$ using $\mathcal{O}\big(\log(n/\rho)\big)$ queries to U and $U^\dagger$, such that with probability $\ge 1 - \rho$, we have*

$$f(y) \ge f(0) + \left\langle \tilde{g}, y\right\rangle - 23^2 \sqrt{\frac{\delta n^3 L}{\rho r_1}} \left\|y\right\|_1 - 2L\sqrt{n} r_1 \qquad \textit{for all } y \in C,$$

[2] The proof of this lemma assumes that the query access is deterministic, however using Corollary 29 of [AGGW18] instead of Lemma 2.6 shows that a relational quantum oracle also suffices as input. In that setting access to $U^\dagger$ should also be assumed.

and hence (by Cauchy-Schwarz)

$$f(y) \geq f(0) + \langle \tilde{g}, y \rangle - (23n)^2 \sqrt{\frac{\delta L}{\rho r_1}} \|y\| - 2L\sqrt{n} r_1 \qquad \textit{for all } y \in C.$$

Proof. Let $r_2 := \sqrt{\frac{\delta r_1 \rho}{nL}}$ and note that $r_2 \leq r_1$. The quantum algorithm works roughly as follows. It first picks a uniformly[3] random $z \in B_\infty(0, r_1)$. Then it uses Jordan's quantum algorithm to compute an approximate gradient at z by approximately evaluating f in superposition over a discrete hypergrid of $B_\infty(z, r_2/n)$. This then yields an approximate subgradient of f at 0.

We now work out this rough idea. Since $B_\infty(z, r_2/n) \subseteq B_1(z, r_2)$, Lemma 3.12 implies

$$\sup_{y \in B_\infty(0, r_2/n)} \left| f(z+y) - f(z) - \langle y, \nabla^{(r_2)} f(z) \rangle \right| \leq \frac{r_2^2 \Delta^{(r_2)} f(z)}{2}. \tag{3.1}$$

Also as shown by Lemma 3.9 and Markov's inequality we have

$$\Delta^{(r_2)} f(z) \leq \frac{2nL}{\rho r_1} \tag{3.2}$$

with probability $\geq 1 - \rho/2$ over the choice of z. If z is such that Equation (3.2) holds, then we get

$$\sup_{y \in B_\infty(0, r_2/n)} \left| f(z+y) - f(z) - \langle y, \nabla^{(r_2)} f(z) \rangle \right| \leq \frac{nLr_2^2}{\rho r_1} = \delta.$$

Now apply the quantum algorithm of Lemma 2.6 with $r = 2r_2/n$, $c = f(z)$, $g = \nabla^{(r_2)} f(z)$, and $B = Lr$. This uses $\mathcal{O}(\log(n/\rho))$ queries to U and $U^\dagger$, and with probability $\geq 1 - \rho/2$ computes an approximate gradient $\tilde{g}$ such that

$$\left\| \nabla^{(r_2)} f(z) - \tilde{g} \right\|_\infty \leq \frac{8 \cdot 42 \pi n}{2r_2} \cdot \delta = 4 \cdot 42 \cdot \pi \sqrt{\frac{\delta n^3 L}{\rho r_1}}. \tag{3.3}$$

Also, if z is such that Equation (3.2) holds, then by Lemma 3.8 we get that

$$\sup_{g \in \underline{\partial} f(z)} \left\| \nabla^{(r_2)} f(z) - g \right\|_1 \leq \frac{r_2 \Delta^{(r_2)} f(z)}{2} \leq \frac{nLr_2}{\rho r_1} = \sqrt{\frac{\delta nL}{\rho r_1}},$$

and therefore by the triangle inequality and Equation (3.3) we get that

$$\begin{aligned}
\sup_{g \in \underline{\partial} f(z)} \|g - \tilde{g}\|_\infty &\leq \sup_{g \in \underline{\partial} f(z)} \left\| g - \nabla^{(r_2)} f(z) \right\|_\infty + \left\| \nabla^{(r_2)} f(z) - \tilde{g} \right\|_\infty \\
&\leq \sup_{g \in \underline{\partial} f(z)} \left\| g - \nabla^{(r_2)} f(z) \right\|_1 + \left\| \nabla^{(r_2)} f(z) - \tilde{g} \right\|_\infty \\
&\leq \sqrt{\frac{\delta nL}{\rho r_1}} + 4 \cdot 42 \cdot \pi \sqrt{\frac{\delta n^3 L}{\rho r_1}} \quad < 23^2 \sqrt{\frac{\delta n^3 L}{\rho r_1}}.
\end{aligned}$$

[3] A discrete quantum computer strictly speaking cannot do this, but (as noted after Lemma 3.9) a uniformly random point from a fine-enough hypergrid suffices.

Thus with probability at least $1-\rho$, for all $y \in C$ and for all $g \in \underline{\partial} f(z)$ we have that

$$\begin{aligned} f(y) &\geq f(z) + \langle g, y-z \rangle \\ &= f(0) + \langle \tilde{g}, y \rangle + \langle g - \tilde{g}, y \rangle + (f(z) - f(0)) + \langle g, -z \rangle \\ &\geq f(0) + \langle \tilde{g}, y \rangle - |\langle g - \tilde{g}, y \rangle| - L\|z\| - \|g\| \|z\| \\ &\geq f(0) + \langle \tilde{g}, y \rangle - \|g - \tilde{g}\|_\infty \|y\|_1 - L\sqrt{n} r_1 - L\sqrt{n} r_1 && \text{(by (2.2))} \\ &\geq f(0) + \langle \tilde{g}, y \rangle - 23^2 \sqrt{\frac{\delta n^3 L}{\rho r_1}} \|y\|_1 - 2L\sqrt{n} r_1 \\ &\geq f(0) + \langle \tilde{g}, y \rangle - (23n)^2 \sqrt{\frac{\delta L}{\rho r_1}} \|y\| - 2L\sqrt{n} r_1. \end{aligned}$$

□

3.4 | Implementing SEP using queries to MEM

Let $K \subseteq \mathbb{R}^n$ be a convex set such that $B(0,r) \subseteq K \subseteq B(0,R)$. Given a membership oracle[4] $\mathrm{MEM}_{\varepsilon,0}(K)$ as in Definition 3.1, we will construct a separation oracle $\mathrm{SEP}_{\eta,\rho}(K)$ as in Definition 3.2. Let x be the point we want to separate from K. We first make a membership query to x itself, receiving answer $x \in B(K,\varepsilon)$ or $x \notin B(K,-\varepsilon)$. Suppose $x \notin B(K,-\varepsilon)$, then we need to find a hyperplane that approximately separates x from K. Due to the rotational symmetry of the separation problem, for ease of notation we assume that $x = -\|x\| e_n$.[5] For this x define $h : \mathbb{R}^{n-1} \to \mathbb{R} \cup \{\infty\}$ as

$$h(y) := \inf_{(y,y_n) \in K} y_n,$$

see also Figure 3.2. Note that although h does not explicitly seem to depend on x, it does depend on x implicitly since we have rotated the space such that $x = -\|x\| e_n$.

[4] For simplicity we assume throughout this section that the membership oracle succeeds with certainty (i.e., its error probability is 0). This is easy to justify: suppose we have a classical T-query algorithm, which uses $\mathrm{MEM}_{\varepsilon,0}(K)$ queries and succeeds with probability at least $1-\rho$. If we are given access to a $\mathrm{MEM}_{\varepsilon,\frac{1}{3}}(K)$ oracle instead, then we can create a $\mathrm{MEM}_{\varepsilon,\frac{\rho}{T}}(K)$ oracle by $\mathcal{O}\left(\log\left(T/\rho\right)\right)$ queries to $\mathrm{MEM}_{\varepsilon,\frac{1}{3}}(K)$ and taking the majority of the answers. Then running the original algorithm with $\mathrm{MEM}_{\varepsilon,\frac{\rho}{T}}(K)$ will fail with probability at most 2ρ. Therefore the assumption of a membership oracle with error probability 0 can be removed at the expense of only a small logarithmic overhead in the number of queries. A similar argument works for the quantum case.

[5] For the query complexity this is without loss of generality, since we can always apply a rotation to all the points such that this holds. If we instead consider the computational cost of our algorithm, then we have to take into account the cost of this rotation and its inverse. Note, however, that this rotation can always be written as the product of n rotations on only 2 coordinates, and hence can be applied in $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ additional steps. These rotations can also be found in $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ time via a greedy algorithm: first find a rotation on coordinates n and $n-1$ that leaves coordinate $n-1$ zero, then similar for coordinates $n-2$ and n, and so on.

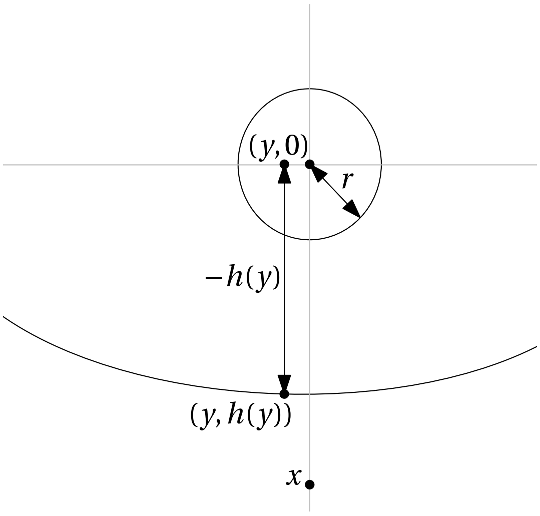


Figure 3.2: Graphical example of the relation between $h(y)$ and the distance from $(y,0)$ to the border in the $-e_n$ direction.

Our h is a bit different from the one used in [LSV18], but we can show that it has many of the same properties. Since K is a convex set, h is a convex function over $\mathbb{R}^{n-1}$. As we show below, the function h is also Lipschitz (Lemma 3.14) and we can approximately compute its value using binary search with $\widetilde{\mathcal{O}}_{\frac{Rn}{r\varepsilon}}(1)$ classical queries to a membership oracle (Lemma 3.15). Furthermore, an approximate subgradient of h at 0 allows to construct a hyperplane approximately separating x from K (Lemma 3.16). Combined with the results of Section 3.3 this leads to the main results of this section, Theorem 3.17 and Theorem 3.18, which show how to efficiently construct a separation oracle using respectively classical and quantum queries to a membership oracle.

Analogously to [LSV18, Lem. 12] we first show that our h is Lipschitz.

Lemma 3.14 *For every $\delta \in (0,r)$, h is $\frac{R}{r-\delta}$-Lipschitz on $B(0,\delta) \subseteq \mathbb{R}^{n-1}$, that is, we have*

$$\left|h(y') - h(y)\right| \le \frac{R}{r-\delta}\left\|y'-y\right\| \quad \textit{for all } y, y' \in B(0,\delta).$$

Proof. Observe that for all $y \in B(0,r)$ we have $-R \le h(y)$, because $B(0,r) \subseteq K \subseteq B(0,R)$, and we have $h(y) \le 0$ because $(y,0) \in K$. Let $y, y' \in B(0,\delta)$ be arbitrary but distinct points. Due to symmetry it will suffice to show that $h(y') - h(y) \le \frac{R}{r-\delta}\left\|y'-y\right\|$.

We will restrict our attention to the line through y and y', i.e., the line given by $y + \lambda z$ for $z := \frac{y'-y}{\|y'-y\|}$ the unit vector corresponding to the direction of the line. Define the point

$$p := y + \left(\left\|y'-y\right\| + (r-\delta)\right)z = y' + (r-\delta)z$$

on this line and note that $p \in B(0,r)$. Since y' lies between y and p on the line it is a convex combination of these two points. In particular, since $\left\|p-y'\right\| = r-\delta$, it is the convex combination

$$y' = \frac{\|y'-y\|}{\|y'-y\|+(r-\delta)} p + \frac{r-\delta}{\|y'-y\|+(r-\delta)} y.$$

Due to convexity

$$h(y') \le \frac{\|y'-y\|}{\|y'-y\|+(r-\delta)} h(p) + \frac{r-\delta}{\|y'-y\|+(r-\delta)} h(y),$$

which implies

$$h(y') - h(y) \le \frac{\|y'-y\|}{\|y'-y\|+(r-\delta)} \big(h(p)-h(y)\big) \le \frac{\|y'-y\|}{r-\delta} R. \qquad \square$$

Now we show how to compute the value of h using membership queries to K.

Lemma 3.15 *For all* $y \in B\big(0,\frac{r}{2}\big) \subset \mathbb{R}^{n-1}$ *we can compute a* δ*-approximation of* $h(y)$ *with* $\mathcal{O}\big(\log\big(\frac{R}{\delta}\big)\big)$ *queries to a* $\mathrm{MEM}_{\varepsilon,0}(K)$ *oracle, where* $\varepsilon \le \frac{r}{3R}\delta$.

Proof. Let $y \in B(0,\frac{r}{2})$, then $(y,h(y))$ is a boundary point of K by the definition of h. Note that $h(y) \in [-R,-r/2]$. Our goal is to perform binary search over this interval to find a good approximation of $h(y)$. If we had access to a perfect membership oracle then we would be done. However, since our membership oracle can give back a wrong answer when queried with a point that is ε-close to the boundary of K, a more careful analysis is needed.

Suppose $y_n \le -\frac{r}{2}$ is our current binary search guess for $h(y)$. We first show that

(*a*) if $(y,y_n) \in B(K,\varepsilon)$, then $y_n \ge h(y) - \delta$, and

(*b*) if $(y,y_n) \notin B(K,-\varepsilon)$, then $y_n \le h(y) + \frac{2}{3}\delta$.

For the proof of (*a*) consider a $g \in \underline{\partial} h(y)$. Since g is a subgradient we have that $h(z) \ge h(y) + \langle g, z-y\rangle$ for all $z \in \mathbb{R}^{n-1}$. Hence, for all $z \in \mathbb{R}^{n-1}$ and z_n such that $(z,z_n) \in K$ we have

$$\left\langle \begin{pmatrix} -g \\ 1 \end{pmatrix}, \begin{pmatrix} y \\ h(y) \end{pmatrix} \right\rangle \le \left\langle \begin{pmatrix} -g \\ 1 \end{pmatrix}, \begin{pmatrix} z \\ h(z) \end{pmatrix} \right\rangle \le \left\langle \begin{pmatrix} -g \\ 1 \end{pmatrix}, \begin{pmatrix} z \\ z_n \end{pmatrix} \right\rangle$$

where the first inequality is a rewriting of the subgradient inequality and the second inequality uses that $z_n \ge h(z)$ since $(z,z_n) \in K$. Since $(y,y_n) \in B(K,\varepsilon)$ it follows from the above inequality that

$$\left\langle \begin{pmatrix} -g \\ 1 \end{pmatrix}, \begin{pmatrix} y \\ y_n \end{pmatrix} \right\rangle \ge \left\langle \begin{pmatrix} -g \\ 1 \end{pmatrix}, \begin{pmatrix} y \\ h(y) \end{pmatrix} \right\rangle - \varepsilon \left\| \begin{pmatrix} -g \\ 1 \end{pmatrix} \right\| \ge \left\langle \begin{pmatrix} -g \\ 1 \end{pmatrix}, \begin{pmatrix} y \\ h(y) \end{pmatrix} \right\rangle - \varepsilon(\|g\|+1).$$

Lemma 3.14 together with the argument of Equation (2.2) implies that $\|g\| \le \frac{2R}{r}$. Since

$$\varepsilon(\|g\|+1) \le \varepsilon\left(\frac{2R}{r}+1\right) \le \varepsilon\frac{3R}{r} \le \delta,$$

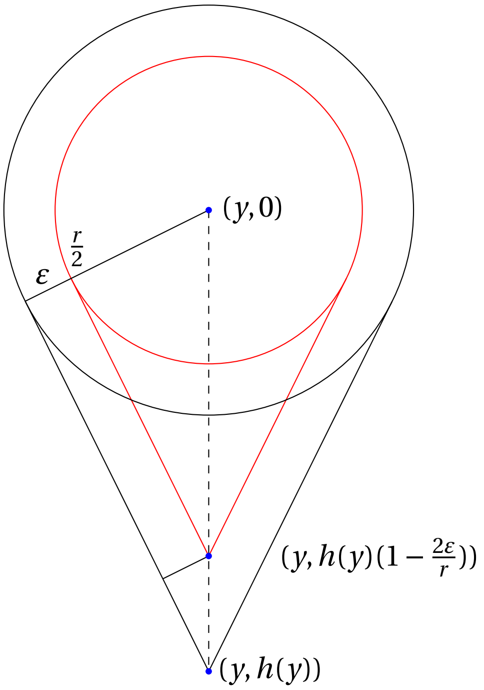


Figure 3.3: Let C be the convex hull of $B((y,0),r/2)$ and $(y,h(y))$, drawn in the figure using a black line. The region $B(C,-\varepsilon)$ is drawn with a red line. Due to similar triangles the hypotenuse of the small triangle at the bottom has length $\frac{2\varepsilon}{r}h(y)$. It follows that $B(C,-\varepsilon)$ is the convex hull of $B((y,0),r/2-\varepsilon)$ and $\left(y,h(y)\left(1-\frac{2\varepsilon}{r}\right)\right)$.

we obtain the inequality of (a).

For (b), consider the convex set C which is the convex hull of $B((y,0),r/2)$ and $(y,h(y))$. From Figure 3.3 it follows that $B(C,-\varepsilon)$ is the convex hull of $B((y,0),r/2-\varepsilon)$ and $\left(y,h(y)\left(1-\frac{2\varepsilon}{r}\right)\right)$. Since $C\subseteq K$, we have $B(C,-\varepsilon)\subseteq B(K,-\varepsilon)$. Therefore $(y,y_n)\not\in B(K,-\varepsilon)$ implies $(y,y_n)\notin B(C,-\varepsilon)$, and

$$y_n\leq h(y)\left(1-\frac{2\varepsilon}{r}\right)=h(y)-\varepsilon\frac{2h(y)}{r}\leq h(y)+\varepsilon\frac{2R}{r}\leq h(y)+\frac{2}{3}\delta.$$

Now we can analyze the binary search algorithm. By making $\mathcal{O}\left(\log\left(\frac{R}{\delta}\right)\right)$ $\mathrm{MEM}_{\varepsilon,0}(K)$ queries to points of the form (y,y_n), we can find a value $y_n\in[-R,-\frac{r}{2}]$ such that $(y,y_n)\in B(K,\varepsilon)$ but $(y,y_n-\frac{\delta}{3})\not\in B(K,-\varepsilon)$. By (a) and (b) together we get that $\left|h(y)-y_n\right|\leq\delta$. □

The following lemma shows how to convert an approximate subgradient of h to a hyperplane that approximately separates x from K.

Lemma 3.16 *Suppose $-\|x\|e_n=x\notin B(K,-\varepsilon)$, and $\tilde{g}\in\mathbb{R}^{n-1}$ is an approximate subgradient of h at 0, meaning that for some $a,b\in\mathbb{R}$ and for all $y\in\mathbb{R}^{n-1}$*

$$h(y)\geq h(0)+\langle\tilde{g},y\rangle-a\|y\|-b,$$

then $s:=\frac{(-\tilde{g},1)}{\|(-\tilde{g},1)\|}$ satisfies $\langle s,z\rangle\geq\langle s,x\rangle-\frac{aR+b}{\|(-\tilde{g},1)\|}-\frac{2R}{r}\frac{\varepsilon}{\|(-\tilde{g},1)\|}$ for all $z\in K$.

Proof. Let us introduce the notation $z = (y, z_n)$ and $s' := (-\tilde{g}, 1) = \|(-\tilde{g}, 1)\| s$, then

$$\begin{aligned}\langle s', z\rangle &= z_n - \langle \tilde{g}, y\rangle \\ &\geq h(y) - \langle \tilde{g}, y\rangle \\ &\geq h(0) - a\|y\| - b \\ &\geq -\|x\| - \frac{2R}{r}\varepsilon - aR - b \\ &= \langle s', x\rangle - aR - b - \frac{2R}{r}\varepsilon,\end{aligned}$$

where the last inequality used claim (b) from the proof of Lemma 3.15 with the point $(0, -\|x\|)$ and $\delta = \frac{3R}{r}\varepsilon$. □

We now construct a separation oracle using $\widetilde{\mathcal{O}}(n)$ classical queries to a membership oracle. In particular, to construct an η-precise separation oracle, we require an ε-precise membership oracle with

$$\varepsilon = \frac{\eta}{676} n^{-2} \left(\frac{r}{R}\right)^3 \left(\frac{\eta}{R}\right)^2 \rho.$$

The analogous result in [LSV18, Thm. 14] uses the stronger assumption[6]

$$\varepsilon \approx \frac{\eta}{8 \cdot 10^6} n^{-\frac{7}{2}} \left(\frac{r}{R}\right)^6 \left(\frac{\eta}{R}\right)^2 \rho^3.$$

Compared to this, our result scales better in terms of n, $\frac{r}{R}$ and ρ.

Theorem 3.17 *Let K be a convex set satisfying $B(0,r) \subseteq K \subseteq B(0,R)$. For any $\eta \in (0,R]$ and $\rho \in (0,1/3]$, we can implement the oracle $\mathrm{SEP}_{\eta,\rho}(K)$ using $\mathcal{O}\left(n \log\left(\frac{n}{\rho}\frac{R}{\eta}\frac{R}{r}\right)\right)$ classical queries to a $\mathrm{MEM}_{\varepsilon,0}(K)$ oracle, where $\varepsilon \leq \eta(26n)^{-2}\left(\frac{r}{R}\right)^3\left(\frac{\eta}{R}\right)^2\rho$.*

Proof. Let $x \notin B(K, -\varepsilon)$ be the point we want to separate from K. Let $\delta := \eta\frac{n^{-2}}{9\cdot 24}\left(\frac{r}{R}\cdot\frac{\eta}{R}\right)^2\rho$, then $\varepsilon \leq \frac{r}{3R}\delta$. By Lemma 3.15 we can evaluate h to within error δ using $\mathcal{O}\left(\log\left(\frac{R}{\delta}\right)\right)$ queries to a $\mathrm{MEM}_{\varepsilon,0}(K)$ oracle. Let us choose $r_1 := \frac{r}{12\sqrt{n}}\frac{\eta}{R}$, then $r_1\sqrt{n} \leq \frac{r}{4}$, therefore $B_\infty(0, 2r_1) \subseteq B(0, r/2)$. Also note that $\delta \leq \frac{\eta}{6\rho} = \frac{2r_1\sqrt{n}R}{\rho r}$. By Lemma 3.14 we know that h is $\frac{2R}{r}$-Lipschitz on $B(0, r/2)$. Hence by Lemma 3.10, using $\mathcal{O}(n)$ queries to h, and therefore $\mathcal{O}\left(n\log\left(\frac{R}{\delta}\right)\right)$ queries to a $\mathrm{MEM}_{\varepsilon,0}(K)$ oracle, we can compute an approximate subgradient $\tilde{g}$ such that with probability at least $1-\rho$ we have

$$h(y) \geq h(0) + \langle \tilde{g}, y\rangle - \frac{3n^{\frac{3}{4}}}{2}\sqrt{\frac{\delta 2R}{\rho r_1 r}}\|y\| - \frac{4R}{r}\sqrt{n}r_1 \qquad \text{for all } y \in \mathbb{R}^{n-1}.$$

Substituting the value of r_1 and δ we get $h(y) \geq h(0) + \langle \tilde{g}, y\rangle - \frac{\eta}{2R}\|y\| - \frac{\eta}{3}$, which by Lemma 3.16 gives an s such that $\langle s, z\rangle \geq \langle s, x\rangle - \frac{5}{6}\eta - \frac{2R}{r}\varepsilon \geq \langle s, x\rangle - \eta$ for all $z \in K$ □

[6] It seems that Lee et al. [LSV18, Alg. 1] did not take into account the change in precision analogous to our Lemma 3.15, therefore one would probably need to worsen their exponent of $\frac{r}{R}$ from 6 to 7.

Finally, we give a proof of our main result: we construct a separation oracle using $\widetilde{\mathcal{O}}(1)$ quantum queries to a membership oracle.

Theorem 3.18 *Let K be a convex set satisfying $B(0,r) \subseteq K \subseteq B(0,R)$. For any $\eta \in (0,R]$ and $\rho \in (0,1/3]$, we can implement the oracle $\mathrm{SEP}_{\eta,\rho}(K)$ using $\mathcal{O}\left(\log\left(\frac{n}{\rho}\right)\log\left(\frac{n}{\rho}\frac{R}{\eta}\frac{R}{r}\right)\right)$ quantum queries to a $\mathrm{MEM}_{\varepsilon,0}(K)$ oracle, where $\varepsilon \le \eta(58n)^{-\frac{9}{2}}\left(\frac{r}{R}\right)^3\left(\frac{\eta}{R}\right)^2\rho$.*

Proof. Let $x \not\in B(K,-\varepsilon)$ be the point we want to separate from K. Furthermore, let $\delta := \eta\frac{23^{-4}}{4\cdot 24}n^{-\frac{9}{2}}\left(\frac{r}{R}\cdot\frac{\eta}{R}\right)^2\rho$, then $\varepsilon \le \frac{r}{3R}\delta$. By Lemma 3.15 we can evaluate h to within error δ using $\mathcal{O}\left(\log\left(\frac{R}{\delta}\right)\right)$ queries to a $\mathrm{MEM}_{\varepsilon,0}(K)$ oracle. Let us choose $r_1 := \frac{r}{12\sqrt{n}}\frac{\eta}{R}$, then $r_1\sqrt{n} \le \frac{r}{4}$, therefore $B_\infty(0,2r_1) \subseteq B(0,r/2)$. Also note that $\delta \le \frac{\eta}{6\rho} = \frac{2r_1 nR}{\rho r}$. By Lemma 3.14 we know that h is $\frac{2R}{r}$-Lipschitz on $B(0,r/2)$. Hence by Lemma 3.13, using $\mathcal{O}\left(\log\left(\frac{n}{\rho}\right)\right)$ queries to h, and therefore $\mathcal{O}\left(\log\left(\frac{n}{\rho}\right)\log\left(\frac{R}{\delta}\right)\right)$ queries to a $\mathrm{MEM}_{\varepsilon,0}(K)$ oracle, we can compute an approximate subgradient $\tilde{g}$ such that with probability at least $1-\rho$ we have

$$h(y) \ge h(0) + \langle \tilde{g}, y\rangle - (23n)^2\sqrt{\frac{2\delta R}{\rho r_1 r}}\|y\| - \frac{4R}{r}\sqrt{n}r_1 \qquad \text{for all } y \in \mathbb{R}^{n-1}.$$

Substituting the value of r_1 and δ we get $h(y) \ge h(0) + \langle \tilde{g}, y\rangle - \frac{\eta}{2R}\|y\| - \frac{\eta}{3}$, which by Lemma 3.16 gives an s such that $\langle s,z\rangle \ge \langle s,x\rangle - \frac{5}{6}\eta - \frac{2R}{r}\varepsilon \ge \langle s,x\rangle - \eta$ for all $z \in K$. □

3.5 | Consequences of convex polarity

Here we justify the central symmetry of Figure 3.1 using the results of Grötschel, Lovász, and Schrijver [GLS88, Sec. 4.4]. We first need to recall the definition and some basic properties of the polar K^* of a set $K \subseteq \mathbb{R}^n$. This is the closed convex set defined as follows:

$$K^* = \left\{y \in \mathbb{R}^n : \langle y, x\rangle \le 1 \text{ for all } x \in K\right\}.$$

It is straightforward to verify that if $B(0,r) \subseteq K \subseteq B(0,R)$, then $B(0,1/R) \subseteq K^* \subseteq B(0,1/r)$, moreover $(K^*)^* = K$ if K is a closed convex set.[7] For the remainder of this section we assume that K is a closed convex set such that $B(0,r) \subseteq K \subseteq B(0,R)$.

We will observe that for the polar K^* of a set K the following holds:

$$\mathrm{MEM}(K^*) \leftrightarrow \mathrm{VAL}(K), \qquad \mathrm{SEP}(K^*) \leftrightarrow \mathrm{VIOL}(K), \tag{3.4}$$

where $\mathrm{MEM}(K^*) \leftrightarrow \mathrm{VAL}(K)$ means we can implement a weak validity oracle for K using a single query to a weak membership oracle for K^*, and vice versa. Since $\mathrm{VIOL}(K)$ and $\mathrm{OPT}(K)$ are equivalent up to reductions that use $\widetilde{\Theta}(1)$ queries (via binary search),

[7] Note that K^* is a dual representation of the convex set K. Each point in K^* corresponds to a (normalized) valid inequality for K. This duality is not to be confused with Lagrangian duality.

this justifies the central symmetry of Figure 3.1, because it shows that algorithms that implement VIOL(K) given VAL(K) are equivalent to algorithms that implement SEP(K^*) given MEM(K^*), and similarly algorithms that implement SEP(K) given VIOL(K) are equivalent to algorithms that implement VIOL(K^*) given SEP(K^*).

Grötschel, Lovász, and Schrijver [GLS88, Sec. 4.4] showed that the weak membership problem for K^* can be solved using a single query to a weak validity oracle for K, and that the weak separation problem for K^* can be solved using a single query to a weak violation oracle for K. Using similar arguments one can show the reverse directions as well, which justifies (3.4). Here we only motivate the equivalences between the above-mentioned weak oracles by showing the equivalence of the strong oracles (i.e., where ρ and ε are 0).

Strong membership on K^* is equivalent to strong validity on K First, for a given vector $c \in \mathbb{R}^n$ and a $\gamma > 0$ observe the following:

$$\frac{c}{\gamma} \not\in \operatorname{int}(K^*) \quad \Longleftrightarrow \quad \exists y \in K \text{ s.t. } \langle c/\gamma, y\rangle \geq 1 \quad \Longleftrightarrow \quad \exists y \in K \text{ s.t. } \langle c, y\rangle \geq \gamma.$$

Hence, a strong membership query to K^* with a point c can be implemented by querying a strong validity oracle for K with the vector c and the value 1. Likewise, a strong validity query to K with a point c and value[8] $\gamma > 0$ can be implemented using a strong membership query to K^* with c/γ.

Strong separation on K^* is equivalent to strong violation on K To implement a strong separation query on K^* for a vector $y \in \mathbb{R}^n$ we do the following. Query the strong violation oracle for K with y and the value 1. If the answer is that $\langle y, x\rangle \leq 1$ for all $x \in K$, then $y \in K^*$. If instead we are given a vector $x \in K$ with $\langle y, x\rangle \geq 1$, then x separates y from K^* (indeed, for all $z \in K^*$, we have $\langle z, x\rangle \leq 1 \leq \langle y, x\rangle$).

For the reverse direction, to implement a strong violation oracle for K on the vector c and value[8] $\gamma > 0$ we do the following. Query the strong separation oracle for K^* with the point c/γ. If the answer is that $c/\gamma \in K^*$ then $\langle c, x\rangle \leq \gamma$ for all $x \in K$. If instead we are given a non-zero vector $y \in \mathbb{R}^n$ that satisfies $\langle c/\gamma, y\rangle \geq \langle z, y\rangle$ for all $z \in K^*$, then $\tilde{y} = y/\langle c/\gamma, y\rangle$ will be a valid answer for the strong violation oracle for K. Indeed, we have $\tilde{y} \in K$ because $\langle z, \tilde{y}\rangle \leq 1$ for all $z \in K^*$ and $K = (K^*)^*$, and by construction $\langle c, \tilde{y}\rangle = \gamma$.

3.6 | Discussion and future work

We mention several open problems for future work:

- Our current implementation of an optimization query using $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ quantum membership queries is quadratically better than the best known classical randomized algorithm, which uses roughly n^2 membership queries. However, to the

[8] Observe that validity and violation queries with value $\gamma \leq 0$ can be answered trivially, since $0 \in K$.

best of our knowledge it is open whether this quadratic classical bound is optimal (a quadratic classical lower bound is known for *deterministic* algorithms [Yao75]).

- Are there interesting convex optimization problems where separation is much harder than membership for classical computers? Such problems would be good candidates for quantum speedup in optimization in the real, non-oracle setting of time complexity. It is known that given a deterministic algorithm for a function, an algorithm with roughly the same complexity can be constructed to compute the gradient of that function [GW08], so for deterministic oracles separation is not much harder than membership queries. This, however, still leaves randomized and quantum membership oracles to be considered.

- The algorithms that give an $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ upper bound on the number of separation queries for optimization (for example [LSW15, Thm. 42]) give the best theoretical results for many convex optimization problems. However, due to the large constants in these algorithms they are rarely used in a practical setting. A natural question is whether the algorithms used in practice lend themselves to quantum speedups.

Chapter 4

Quantum algorithms for SDP-solving

In this chapter we consider *semidefinite programs* (SDPs). As first observed by Brandão and Svore [BS17], the intermediate solutions of certain classical SDP-solving frameworks can be stored as quantum states on a logarithmic number of qubits, allowing for a quadratic speedup in terms of the dimension n and the number of constraints m of the SDP. We start by refining their techniques to bring down their original complexity of $\widetilde{\mathcal{O}}\left(\sqrt{nm}s^2\gamma^{32}\right)$ (where s is the row and column sparsity of the input matrices and γ is the inverse of a scale-invariant error parameter) to $\widetilde{\mathcal{O}}\left(\sqrt{nm}s^2\gamma^8\right)$. We then introduce multiple new techniques and ideas to bring down the complexity further and allow for new input models as well. This results in a complexity of $\widetilde{\mathcal{O}}\left(\left(\sqrt{m}+\sqrt{n}\gamma\right)\alpha\gamma^4\right)$ in the more general *quantum operator input model*, where $\alpha = s$ for the classical sparse matrix input model.

We finish the chapter by applying our SDP-solvers to a few different problems. For the problem of *shadow tomography*, where we are asked to approximate the expectation values of a set of measurements on a quantum state, we give an improvement in both the number of required samples and the computational complexity. For the problem of *quantum state discrimination*, where we need to find the optimal measurement to distinguish a set of states, and the problem of *optimal design*, where we need to design an optimal experiment, we give algorithms that have a quadratic speedup in some parameters at the cost of a large polynomial complexity in other parameters. These results show that there is the possibility for a speedup here, and reducing the dependence on the other parameters is an interesting direction for further research.

This chapter is based on the papers “Quantum SDP-Solvers: Better upper and lower bounds” by J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf [AGGW17] and “Improvements in Quantum SDP-Solving with Applications” by J. van Apeldoorn and A. Gilyén [AG19a].

4.1 | Introduction

In this chapter we consider quantum algorithms for semidefinite programs. We consider the basic (primal) form of an SDP as introduced in Section 2.4.3:

$$\begin{aligned} \max_{X \in \mathbb{R}^{n\times n}} \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}\big(A_j X\big) \leq b_j \quad \text{for all } j \in [m] \\ & X \succeq 0, \end{aligned} \tag{4.1}$$

with dual problem

$$\begin{aligned} \min \quad & b^T y \\ \text{s.t.} \quad & \sum_{j=1}^{m} y_j A_j - C \succeq 0 \\ & y \geq 0. \end{aligned} \tag{4.2}$$

The input to the problem consists of $n \times n$ Hermitian constraint matrices $A_1, \ldots, A_m$, an objective matrix C, and reals $b_1, \ldots, b_m$. For normalization purposes we assume $\|C\|, \|A_j\| \leq 1$. The number of constraints is m (we do not count the standard $X \succeq 0$ constraint for this). The variable X of this SDP is an $n \times n$ positive semidefinite (psd) matrix. We assume that $A_1 = I$ and $b_1 = R$, giving a known bound on the trace of a solution: $\mathrm{Tr}(X) \leq R$. We assume that the dual optimum is attained and that an explicit $r \geq 1$ is known such that at least one optimal dual solution y exists with $\|y\|_1 \leq r$. These assumptions imply that strong duality holds, justifying the use of OPT for both the optimal value of the primal SDP and for the optimal value of the dual SDP. The goal will be to find an additive ε-approximation of OPT, and a description of an (almost) feasible point for the SDP that attains this value (see Section 4.1.1 for a formal definition). We assume that both R and r are at least 1 and that $\varepsilon \leq 1/6$.

Classical SDP-solvers Ever since Dantzig's development of the simplex algorithm for solving LPs in the 1940s [Dan48], much work has gone into finding faster solvers, first for LPs and then also for SDPs. The simplex algorithm for LPs (with some reasonable pivot rule) is usually fast in practice, but has worst-case exponential runtime. The ellipsoid method and interior-point methods can solve LPs and SDPs in polynomial time; they will typically *approximate* the optimal value to arbitrary precision [GLS81; NN94]. The best known general SDP-solvers [LSW15][1] approximate the optimal value OPT of such an SDP up to additive error ε, with complexity

$$\mathcal{O}(m(m^2 + n^{2.373} + mns)\,\mathrm{polylog}(m, n, R, 1/\varepsilon)),$$

[1] For LP-solvers better results are known. Recently Cohen, Lee and Song gave an LP-solver with a logarithmic dependence on the error and an $\widetilde{\mathcal{O}}\big(n^{2.373}\big)$ dependence on the number of variables [CLS19]. Here the 2.373 exponent is the currently best known upper bound on the exponent for matrix multiplication; if this exponent is improved to 2, then their algorithm has a complexity that scales as $\widetilde{\mathcal{O}}_{\frac{1}{\varepsilon}}\big(n^{2+1/6}\big)$.

where 2.373 is the currently best known upper bound on the exponent for matrix multiplication; s is the *sparsity*: the maximal number of non-zero entries per row of the input matrices.[2] The assumption here is that the rows and columns of the matrices of SDP (4.1) can be accessed as adjacency lists: we can query, say, the ℓth non-zero entry of the kth row of matrix A_j in constant time.

Arora and Kale [AK16] gave an alternative way to approximate OPT, using a matrix version of the "multiplicative weights update" method.[3] In Section 4.2.1 we will describe their framework in more detail. The framework of Arora and Kale is really a meta-algorithm, it does not specify how to implement a certain subroutine. They themselves provide subroutines that are optimized for special cases, which allows them to give highly optimized upper bounds for solving these specific SDPs. For example, for the MAXCUT SDP they obtain a solver with near-linear runtime in the number of edges of the graph. They also observed that the algorithm can be made more efficient by not explicitly calculating intermediate solutions in each iteration. Arora and Kale do not describe how to solve general SDPs, but it can be shown that one can get a general classical SDP-solver in their framework with complexity

$$\widetilde{\mathcal{O}}\left(nms\left(\frac{Rr}{\varepsilon}\right)^4 + ns\left(\frac{Rr}{\varepsilon}\right)^7\right). \tag{4.3}$$

Compared to the complexity of the SDP-solver of [LSW15], this has much worse dependence on R and ε, but better dependence on m and n. Using the Arora-Kale framework is thus preferable over standard SDP-solvers for the case where Rr is small compared to mn, and a rough approximation to OPT (say, small constant ε) is good enough.

Prior work on quantum SDP-solvers In 2016 Brandão and Svore [BS17] used the Arora-Kale framework to implement a general quantum SDP-solver in the sparse matrix input model. They observed that the $n \times n$-matrix

$$\rho := \frac{e^{-\sum_{j=0}^{m} y_j A_j + y_0 C}}{\mathrm{Tr}\left(e^{-\sum_{j=0}^{m} y_j A_j + y_0 C}\right)}, \tag{4.4}$$

that is used for calculations in the Arora-Kale framework is in fact a $\log(n)$-qubit Gibbs state and can be efficiently prepared as a quantum state on a quantum computer. Using this they achieved a quantum speedup in terms of n. Combining this with a Grover-like speedup allowed for a speedup in terms of m as well, leading to an ε-approximate

[2] See Lee, Sidford, and Wong [LSW15, Sec. 10.2 of arXiv version 2], and note that our m, n are their n, m, their S is our mns, and their M is our R. The bounds for other SDP-solvers that we state later also include another parameter r; it follows from the assumptions of [LSW15, Thm. 45 of arXiv version 2] that in their setting $r \le mR$, and hence r is absorbed in the polylog(mnR/ε) factor.

[3] See also [AHK12] for a subsequent survey; the same algorithm was independently discovered around the same time in the context of learning theory [TRW05; WK12]. In the optimization community, first-order methods for semidefinite programming have been considered for instance in [Ren16; Ren19].

quantum SDP-solver with complexity

$$\widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\left(\frac{Rr}{\varepsilon}\right)^{32}\right).$$

They also showed an $\Omega(\sqrt{m}+\sqrt{n})$ quantum query lower bound for solving SDPs when all other parameters are constant. This left as an open question whether the gap between the upper and lower bound could be closed.

In our first work on quantum SDP-solvers [AGGW17] we improved the error dependence in the upper bound drastically, leading to a total complexity of

$$\widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\left(\frac{Rr}{\varepsilon}\right)^{8}\right).$$

These results are discussed in Section 4.4. We also considered the limitations of these types of SDP-solvers and gave general lower bounds on quantum SDP-solving. This second set of results is part of Chapter 9.

Recently Brandão et al. [BKLLSW17] gave an improved SDP-solver when the input matrices are proportional to quantum states that we can prepare[4] that has a complexity bound with logarithmic dependence on n (where $A_1 = I$ is threaded separately):

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}_n\left(\sqrt{m}\,\mathrm{poly}\left(\frac{Rr}{\varepsilon}, B, \max_{j\in\{2,\ldots,m\}} \mathrm{rank}\big(A_j\big), \mathrm{rank}(C)\right)\right).$$

Here B is an upper bound on the scale difference between the input matrices and the quantum states we can sample (see Section 4.5.1 for a formal definition of this input model). Brandão et al. also applied their algorithm to the problem of *shadow tomography*, giving the first non-trivial application of a quantum SDP-solver.

In the same work these results were further improved by the introduction of the Fast Quantum OR lemma. Approaches prior to [BKLLSW17] searched for a violated constraint in the SDP using Grover-like techniques, resulting in a multiplicative complexity of Gibbs-sampling and searching. The Fast Quantum OR lemma can be used to separate the search phase from the initial Gibbs state preparation phase. This led to the improved complexity bound [BKLLSW17] of

$$\widetilde{\mathcal{O}}_n\left(\left(\sqrt{m} + \mathrm{poly}\left(\max_{j\in\{2,\ldots,m\}} \mathrm{rank}\big(A_j\big), \mathrm{rank}(C)\right)\right)\mathrm{poly}\left(\frac{Rr}{\varepsilon}, B\right)\right).$$

We thank the authors of [BKLLSW17] for sending us an early draft of [BKLLSW17] introducing the Fast Quantum OR Lemma, which enabled us to work on these improvements. During the correspondence [Wu17] the application of the Fast OR lemma to other input models was independently suggested by Brandão et al. and by us.

[4] This model was already introduced in the first version of [BS17] together with a similar complexity statement, but there were some unresolved issues in the proof, that were only fixed by the contributions of [BKLLSW17].

This result appeared both in [BKLLSW19] (a later version of [BKLLSW17]) and in our work [AG19a].

In the same paper we also removed the rank dependence in the setting where the input matrices are proportional to quantum states that we can sample. We also introduced a new input model based on block-encodings that allowed us to speed up SDP-solving in several different models, including the sparse matrix input model.

Our results & overview The rest of this section is used to formalize the definitions of an *SDP-solver*, the input oracles, and the computational model. In Section 4.2 we introduce two meta-algorithms that can be used to solve an SDP. Both these meta-algorithms will still require us to implement a few subroutines. In particular we need to create purifications of states of the form (4.4) and we need to implement a procedure that on input ρ outputs a random variable with expectation value $\mathrm{Tr}(A_j\rho)$. In Section 4.3 we show how to do both these things when we have access to block-encodings of the matrices involved. In Section 4.4 we show how to implement an oracle for the sum $-\sum_{j=0}^m y_jA_j+y_0C$ when the input matrices are given via sparse matrix access. This leads to an SDP-solver that uses

$$\widetilde{\mathcal{O}}\left(\sqrt{nm}s^2\left(\frac{Rr}{\varepsilon}\right)^8\right)$$

queries and elementary operations. In Section 4.5 we consider the more general *quantum operator input model*. In this model the input matrices are given as block-encodings. We start by showing that a few different input models reduce to this model in a natural way. We then show how a block-encoding for the sum $-\sum_{j=0}^m y_jA_j+y_0C$ can be implemented more efficiently than before. This leads to an SDP-solver that uses

$$\widetilde{\mathcal{O}}\left(\sqrt{nm}\alpha\left(\frac{Rr}{\varepsilon}\right)^4\right)$$

queries and elementary operations, where α is a parameter for this new model. For the sparse matrix input model we can take $\alpha=s$. In Section 4.6 we show how the *Fast Quantum OR Lemma* can be used for faster search and minimum-finding techniques. This leads to an SDP-solver that uses

$$\widetilde{\mathcal{O}}\left(\left(\sqrt{m}+\sqrt{n}\left(\frac{Rr}{\varepsilon}\right)\right)\alpha\left(\frac{Rr}{\varepsilon}\right)^4\right)$$

queries and elementary operations. In Section 4.7 we give a few different reductions that show that R, r and $1/\varepsilon$ are equivalent, in the sense that we can always make two of these constant by increasing the third. For this reason we often consider $\left(\frac{Rr}{\varepsilon}\right)$ as a single parameter that we call γ. We finish the chapter in Section 4.8 by giving a few different applications of our SDP-solvers.

4.1.1 | SDP-solvers

We say an algorithm is an ε-approximate *quantum SDP-solver* if for all input numbers $\omega \in \mathbb{R}$ and $\zeta \in (0,1)$, with success probability $1-\zeta$, all of the following hold:

- The algorithm determines whether $\mathrm{OPT} \leq \omega+\varepsilon$ or $\mathrm{OPT} \geq \omega-\varepsilon$. If $\mathrm{OPT} \in [\omega-\varepsilon, \omega+\varepsilon]$, then it may output either.
- The algorithm finds a $y \in \mathbb{R}^m_{\geq 0}$ that is an ε*-feasible solution* to the dual problem with objective value at most $\omega+\varepsilon$, i.e., it finds a y such that

  $$\sum_{j=1}^m y_j A_j - C \succeq -\varepsilon I$$
  $$\langle y, b\rangle \leq \omega + \varepsilon.$$

  Alternatively, it may conclude correctly that no such y exists if we would set $\varepsilon = 0$ in the inequalities above.
- The algorithm finds a vector $y' \in \mathbb{R}^{m+1}_{\geq 0}$ and a non-negative real number z such that for

  $$\rho := \frac{e^{-\sum_{j=1}^m y'_j A_j + y'_0 C}}{\mathrm{Tr}\left(e^{-\sum_{j=1}^m y'_j A_j + y'_0 C}\right)} \tag{4.5}$$

  we have that $z\rho$ is an ε*-feasible primal solution* with objective value at least $\omega-\varepsilon$, i.e., $z\rho$ is such that

  $$\mathrm{Tr}\big(z\rho A_j\big) \leq b_j + \varepsilon \quad \forall j \in [m]$$
  $$\mathrm{Tr}\big(z\rho C\big) \geq \omega - \varepsilon.$$

  Alternatively, it may conclude correctly that no such z and y' exist if we would set $\varepsilon = 0$ in the inequalities above.

For an ε-approximate SDP-solver the relevant scale-invariant parameter is $\gamma := Rr/\varepsilon$. An algorithm that only satisfies the last of the three points above will be called an ε-*approximate SDP primal oracle*. For such an algorithm the relevant scale-invariant parameter is $\gamma := R/\varepsilon$. Due to the form of the objective value constraint in the last point, and to simplify statements like (4.5), we write $A_0 := -C$ and $b_0 := -\omega$.

Notice that we can easily find an approximation of OPT using binary search on ω if we have an ε-approximate SDP-solver. It is important to note that this is not the case for an ε-approximate SDP primal oracle. If an ε-approximate SDP primal oracle outputs an X, then this does not imply that the optimal value is at least $\omega-\varepsilon$, since the solution that was found might not be feasible. Hence, performing binary search using an ε-approximate SDP primal oracle will not always yield OPT with error less than or equal to ε.

However, if we let $\varepsilon' := \varepsilon/2r$, then binary search with an ε'-approximate SDP primal oracle does yield an ε-approximation of OPT. In particular, let X be the output of an ε'-approximate SDP primal oracle, then X is feasible for a primal SDP where the constraints have right-hand sides $b'_i = b_i + \varepsilon'$ and the original left-hand sides. For this relaxed SDP the optimal value is larger than $\omega - \varepsilon'$, as certified by X. The dual of this SDP has the same feasible region as the original SDP (we did not alter C or the A_j's) but has objective $\left\langle b'_j, y \right\rangle = \langle b_j, y \rangle + \varepsilon' \sum_{j=1}^m y_j$. We know that an original dual optimizer y^* exists that is still feasible for the new SDP, and has an ℓ_1-norm upper bounded by r, since the feasible region of the dual did not change. Hence $\left\langle b'_j, y^* \right\rangle \le \langle b, y^* \rangle + \varepsilon' r = \mathrm{OPT} + \varepsilon' r$ where OPT is the optimal value of the original dual SDP. Hence $\mathrm{OPT} \ge \left\langle b'_j, y^* \right\rangle - \varepsilon' r \ge \omega - \varepsilon' r - \varepsilon'$ where the last inequality follows from weak duality for the altered SDP.

4.1.2 | Input oracles & computational model

We will consider two input models: the *sparse matrix input model* and the *quantum operator input model*. The first model is a quantum version of classical sparse matrix access. The second model is an inherently quantum input model based on the idea of block-encodings. This last model generalizes the sparse matrix input model, and in fact many other reasonable model for SDPs, as we will show later.

In all models we assume quantum oracle access to the numbers b_j via a standard binary input oracle O_b satisfying[5] for all $j \in [m]$:

$$O_b |j\rangle|0\rangle = |j\rangle|b_j\rangle.$$

For all input oracles we assume we can apply both the oracle and its inverse[6] in a controlled fashion.

Sparse matrix input model In the *sparse matrix input model* the input matrices are assumed to be s-row sparse for a known bound $s \in [n]$, meaning that there are at most s non-zero elements per row. The model is similar to the classical model for sparse matrices. Access to the A_j matrices is provided by two oracles, similar to previous work on Hamiltonian simulation in [BCK15]. The first of the two oracles is a unitary O_A^{sparse}, which serves the purpose of sparse access. This oracle calculates the $\mathbf{index}: [m] \times [n] \times [s] \to [n]$ function, which for input (j,k,ℓ) gives the column index of the ℓth non-zero element in the kth row of A_j. We assume this oracle computes the index "in place":

$$O_A^{\text{sparse}} |j,k,\ell\rangle = |j,k,\mathbf{index}(j,k,\ell)\rangle \tag{4.6}$$

[5] For simplicity we assume that all the oracles work on at most $\mathcal{O}(\log(nmRr/\varepsilon))$ qubits, and hence that the bitstring representations of the input numbers are at most $\mathcal{O}(\log(nmRr/\varepsilon))$ bits.

[6] When we talk about samples, e.g. in Section 4.8.1, then we do not assume we can apply the inverse operation.

(In the degenerate case where the kth row has fewer than ℓ non-zero entries, we define $\mathbf{index}(j,k,\ell)$ to be ℓ together with some special symbol indicating this case).

We also need another oracle O_A, returning a bitstring[5] representation of $(A_j)_{ki}$ for every $j \in [m]$ and $k,i \in [n]$:

$$O_A|j,k,i,z\rangle = |j,k,i,z \oplus (A_j)_{ki}\rangle. \tag{4.7}$$

This model corresponds directly to a classical way of accessing sparse matrices.

Quantum operator input model Motivated by recent work [LC16; GSLW19] we propose a new input model that we call the *quantum operator input model.* In this model the input matrices are given by a unitary that implements a block-encoding (see Section 2.3.4). In particular let U_j be an $(\alpha, a, 0)$-block-encoding of A_j, for some fixed[7] $\alpha \geq 1$ and $a = \mathcal{O}\big(\log(nmRr/\varepsilon)\big)$. We assume access to a unitary U that acts as

$$U|j\rangle|\psi\rangle = |j\rangle U_j|\psi\rangle.$$

It can be shown that many other input models can be reduced to the quantum operator input model, for example, the sparse matrix input model can be reduced to the quantum operator input model with $\alpha = s$. In Section 4.5.1 we also consider a few other input models and show how these can be reduced to the quantum operator input model. These include when the input is given as quantum states or when the input is given via the ability to evolve under a Hamiltonian. We also show that if we can perform a measurement corresponding to $A_j \succeq 0$ on our quantum computer using a ancilla qubits, i.e., accept a state ρ with probability $\mathrm{Tr}\big(A_j\rho\big)$, then we can implement a $(1, a+1, 0)$-block-encoding of A_j. In this way this input model is a common generalization of many input models for SDPs, since it is reasonable to assume that other input models allow our quantum computer to accept a state ρ with probability $\mathrm{Tr}\big(A_j\rho\big)$.

Computational cost We analyze the query complexity of algorithms and subroutines, i.e., the number of queries to controlled versions of the input oracles and their inverses. We will denote the optimal quantum query complexity of an ε-approximate *quantum SDP-solver* with success probability $2/3$ by $T_{SDP}(\varepsilon)$. We only consider success probability $2/3$ to simplify the notation and proofs. However, in all cases an ε-approximate SDP-solver with success probability $1-\zeta$ can easily be constructed using $\mathcal{O}\big(\log(1/\zeta)T_{SDP}(\varepsilon)\big)$ queries.

In our algorithms we will assume access to a quantum-read/classical-write RAM (QCRAM), and consider one read/write operation as an elementary operation; the size of the QCRAM will typically be $\widetilde{\mathcal{O}}_{nm}\big(\gamma^2\big)$ bits. Most often in our results the number of non-query elementary operations, i.e., two-qubit gates and QCRAM calls, matches the query complexity up to polylog factors. In particular, if not otherwise stated, in our results a T-query quantum algorithm uses at most $\widetilde{\mathcal{O}}_{nm}(T)$ elementary operations.

[7] Having a single normalization parameter α is not a serious restriction as it is easy to make a block-encoding more subnormalized so that every A_j gets the same normalization, cf. Lemma 2.9.

4.2 | Meta-algorithms

In this section we will introduce two different meta-algorithms. These are frameworks for solving SDPs that leave open the implementation of a certain subroutine. The first meta-algorithm we will discuss is the Arora-Kale framework. This framework allows us to find an approximate dual solution and the optimal value for an SDP. The second meta-algorithm is conceptually simpler and allows us to implement an SDP primal oracle. Together these methods implement an approximate SDP-solver when supplied with correct subroutines. Luckily, both the meta-algorithms require very similar subroutines. In this section we will assume that a guess $\omega \in [-R, R]$ for OPT is fixed.

4.2.1 | The Arora-Kale meta-algorithm

In this section we give a short introduction to the Arora-Kale framework for solving semidefinite programs. We refer to [AK16; AHK12] for a more detailed description and omitted proofs.

The key building block is the Matrix Multiplicative Weights (MMW) algorithm introduced by Arora and Kale in [AK16]. The MMW algorithm can be seen as a strategy for you in a game between you and an adversary. We first introduce the game. There is a number of rounds T. In each round you present a density matrix ρ to an adversary, the adversary replies with a *loss matrix* M satisfying $-I \preceq M \preceq I$. After each round you have to pay $\mathrm{Tr}(M\rho)$. Your objective is to pay as little as possible. The MMW algorithm is a strategy for you that allows you to lose not too much, in a sense that is made precise below. In Algorithm 4.1 we state the MMW algorithm, the following theorem shows the key property of the output of the algorithm.

Theorem 4.1 : [AK16, Thm. 3.1] *For every adversary, the sequence $\rho^{(1)}, \ldots, \rho^{(T)}$ of density matrices constructed using the Matrix Multiplicative Weights Algorithm (Algorithm 4.1) satisfies*

$$\sum_{t=1}^{T} \mathrm{Tr}\left(M^{(t)}\rho^{(t)}\right) \leq \lambda_{\min}\left(\sum_{t=1}^{T} M^{(t)}\right) + \eta \sum_{t=1}^{T} \mathrm{Tr}\left((M^{(t)})^2 \rho^{(t)}\right) + \frac{\ln(n)}{\eta}.$$

Arora and Kale use the MMW algorithm to construct an SDP-solver. For that, they construct an adversary who promises to satisfy an additional condition: in each round t, the adversary returns a matrix $M^{(t)}$ whose trace inner product with the density matrix $\rho^{(t)}$ is non-negative. The above theorem shows that then, after T rounds, the average of the adversary's responses satisfies the stronger condition that its smallest eigenvalue is not too negative: $\lambda_{\min}\left(\frac{1}{T}\sum_{t=1}^{T} M^{(t)}\right) \geq -\eta - \frac{\ln(n)}{\eta T}$. More explicitly, given a guess ω for the optimal value, the MMW algorithm is used to build a vector $y \geq 0$ such that

$$\frac{1}{T}\sum_{t=1}^{T} M^{(t)} \propto \sum_{j=1}^{m} y_j A_j - C$$

Input Parameter $\eta \in (0,1]$, number of rounds T.

Rules In each round player 1 (you) presents a density matrix ρ, and player 2 (the adversary) replies with a matrix M satisfying $-I \preceq M \preceq I$.

Output A sequence of symmetric $n \times n$ matrices $M^{(1)}, \ldots, M^{(T)}$ satisfying $-I \preceq M^{(t)} \preceq I$, for $t \in [T]$ and a sequence of $n \times n$ psd matrices $\rho^{(1)}, \ldots, \rho^{(T)}$ satisfying $\mathrm{Tr}\left(\rho^{(t)}\right) = 1$ for $t \in [T]$.

Strategy of player 1**:**

Take $\rho^{(1)} := I/n$.
In round t:

1. Show the density matrix $\rho^{(t)}$ to the adversary.
2. Obtain the loss matrix $M^{(t)}$ from the adversary.
3. Update the density matrix as follows:

$$\rho^{(t+1)} := \frac{e^{-\eta \sum_{\tau=1}^{t} M^{(\tau)}}}{\mathrm{Tr}\left(e^{-\eta \sum_{\tau=1}^{t} M^{(\tau)}}\right)}.$$

Algorithm 4.1: Matrix Multiplicative Weights (MMW) Algorithm

and $b^T y \leq \omega$. That is, the smallest eigenvalue of the matrix $\sum_{j=1}^{m} y_j A_j - C$ is only slightly below zero and y's objective value is at most ω. Since $A_1 = I$, increasing the first coordinate of y makes the smallest eigenvalue of $\sum_j y_j A_j - C$ bigger, so that this matrix becomes psd and hence dual-feasible. By the above we know how much the minimum eigenvalue has to be shifted, and with the right choice of parameters it can be shown that this gives a dual-feasible vector $\overline{y}$ that satisfies $b^T \overline{y} \leq \omega + \varepsilon$. In order to present the algorithm formally, we require some definitions.

Given a candidate solution $X \succeq 0$ for the primal problem (4.1) and a parameter $\varepsilon \geq 0$, define the polytope

$$\begin{aligned} \mathscr{P}_\varepsilon(X) := \{y \in \mathbb{R}^m : {} & b^T y \leq \omega, \\ & \mathrm{Tr}\left(\left(\sum_{j=1}^{m} y_j A_j - C\right) X\right) \geq -\varepsilon, \\ & y \geq 0\}. \end{aligned}$$

One can verify the following:

Lemma 4.2 : [AK16, Lem. 4.2] *If for a given candidate solution $X \succeq 0$ the polytope $\mathscr{P}_0(X)$ is empty, then a scaled version of X is primal-feasible and of objective value at least ω.*

The Arora-Kale framework for solving SDPs uses the MMW algorithm where the role of the adversary is taken by an ε-*approximate advisor*:[8]

Definition 4.3 : Approximate advisor *An ε-*approximate advisor *is an algorithm that as input takes an $n \times n$ psd matrix X, a parameter $\omega \in [-R, R]$, and the description of an SDP as in* (4.1). *The algorithm either outputs a vector y from the polytope $\mathscr{P}_\varepsilon(X)$ or it outputs "fail". It may only output "fail" if $\mathscr{P}_0(X) = \emptyset$. We write* $\mathsf{Advisor}_\varepsilon$ *for such an algorithm.*

As we will see later, the runtime of the Arora-Kale framework depends on a property of the advisor called the *width*:

Definition 4.4 : Width of $\mathsf{Advisor}_\varepsilon$ *The* width *of* $\mathsf{Advisor}_\varepsilon$ *for an SDP is the smallest $w^* \geq 0$ such that for every $X \succeq 0$ and $\omega \in [-R, R]$, the vector y returned by* $\mathsf{Advisor}_\varepsilon$ *satisfies $\left\|\sum_{j=1}^m y_j A_j - C\right\| \leq w^*$.*

In practice, the width of an advisor is not always known. However, it suffices to work with an upper bound $w \geq w^*$: as we can see in Meta-Algorithm 4.1, the purpose of the width is to rescale the matrix $M^{(t)}$ in such a way that it forms a valid response for the adversary in the MMW algorithm. The following theorem shows the correctness of the Arora-Kale primal-dual meta-algorithm for solving SDPs, stated in Meta-Algorithm 4.1:

Theorem 4.5 : [AK16, Thm. 4.7] *Suppose we are given an SDP of the form* (4.1) *with input matrices $A_1 = I, A_2, \ldots, A_m$ and C having operator norm at most* 1, *and input reals $b_1 = R, b_2, \ldots, b_m$. If Meta-Algorithm 4.1 does not output "fail" in any of the rounds, then the returned vector $\overline{y}$ is feasible for the dual* (4.2) *with objective value at most $\omega + \varepsilon$. If* $\mathsf{Advisor}_{\varepsilon/3}$ *outputs "fail" in the t-th round, then a suitably scaled version of $X^{(t)}$ is primal-feasible with objective value at least ω.*

The SDP-solver uses $T = \left\lceil \frac{9w^2R^2\ln(n)}{\varepsilon^2} \right\rceil$ iterations. In each iteration several steps have to be taken. The most expensive two steps are computing the matrix exponential of the matrix $-H^{(t)}$ and the application of the advisor. Note that the only purpose of computing the matrix exponential is to allow the advisor to compute the values $\mathrm{Tr}\left(A_j X\right)$ for all j and $\mathrm{Tr}(CX)$, since the polytope depends on X only through those values (where $X = R\rho$). To obtain faster algorithms it is important to note, as was done already by Arora and Kale, that the primal-dual algorithm also works if we provide a (more accurate) advisor with approximations of $\mathrm{Tr}\left(A_j X\right)$. Let $a_j := \mathrm{Tr}\left(A_j \rho\right) = \mathrm{Tr}\left(A_j X\right)/\mathrm{Tr}(X)$ and $c := \mathrm{Tr}\left(C\rho\right) = \mathrm{Tr}(CX)/\mathrm{Tr}(X)$. Then, given a list of reals $\tilde{a}_1, \ldots, \tilde{a}_m, \tilde{c}$ and a parameter

[8] In the work by Arora and Kale an approximate advisor is called an *approximate oracle*. However, we decided to not use this more standard naming since the term "oracle" might cause confusion with the input oracles or an SDP primal oracle. We want to stress that our naming is not used elsewhere in the literature.

[9] The first coordinate of the vectors that is added to $y^{(t-1)}$ corresponds with the $A_0 = -C$ matrix.

Input The input matrices and reals of SDP (4.1) and trace bound R. The current guess ω of the optimal value of the dual (4.2). An additive error tolerance $\varepsilon > 0$. An $\frac{\varepsilon}{3}$-approximate advisor $\mathsf{Advisor}_{\varepsilon/3}$ as in Definition 4.3 with width-bound w.

Output Either "Lower" and a vector $\overline{y} \in \mathbb{R}^m_+$ feasible for (4.2) with $b^T\overline{y} \leq \omega + \varepsilon$ or "Higher" and a symmetric $n \times n$ matrix X that, when scaled suitably, is primal-feasible with objective value at least ω.

```
$T := \left\lceil \frac{9w^2R^2\ln(n)}{\varepsilon^2} \right\rceil$.
$\eta := \sqrt{\frac{\ln(n)}{T}}$.
$\rho^{(1)} := I/n$.
for $t = 1, \ldots, T$ do
    Run Advisor$_{\varepsilon/3}$ with $X^{(t)} = R\rho^{(t)}$.
    if Advisor$_{\varepsilon/3}$ outputs "fail" then
        return "Higher" and a description of $X^{(t)}$.
    end if
    Let $\hat{y}$ be the vector generated by Advisor$_{\varepsilon/3}$.
    $M^{(t)} := \frac{1}{w}\left(\sum_{j=1}^m \hat{y}_j A_j - C\right)$.
    Update[9] $y^{(t)} := y^{(t-1)} + \frac{\eta}{w}(1, \hat{y})$.
    Define $H^{(t)} := \eta \sum_{\tau=1}^t M^{(\tau)} = \sum_{j=0}^m y_j^{(t)} A_j$.
    Update the state matrix as follows: $\rho^{(t+1)} := \exp\left(-H^{(t)}\right)/\mathrm{Tr}\left(\exp\left(-H^{(t)}\right)\right)$.
end for
If Advisor$_{\varepsilon/3}$ does not output "fail" in any of the $T$ rounds, then output "Lower" and
the dual solution $\overline{y} = \frac{\varepsilon}{R}e_1 + \frac{w}{\eta T}\sum_{t=1}^T y^{(t)}$ (where we drop the 0th coordinate in $y^{(t)}$).
```

Meta-Algorithm 4.1: Primal-Dual Algorithm for solving SDPs.

$\theta \geq 0$, such that $|\tilde{a}_j - a_j| \leq \theta$ for all j, and $|\tilde{c} - c| \leq \theta$, we define the polytope

$$\begin{aligned}\tilde{\mathscr{P}}(\tilde{a}_1, \ldots, \tilde{a}_m, \tilde{c} - (r+1)\theta) := \{y \in \mathbb{R}^m : {} & b^T y \leq \omega, \\ & \sum_{j=1}^m y_j \leq r, \\ & \sum_{j=1}^m \tilde{a}_j y_j \geq \tilde{c} - (r+1)\theta \\ & y \geq 0\}.\end{aligned}$$

For convenience we will denote $\tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_m)$ and $c' := \tilde{c} - (r+1)\theta$. Notice that $\tilde{\mathscr{P}}$ also contains a new type of constraint: $\sum_j y_j \leq r$. Recall that r is defined as a positive real such that there exists an optimal solution y to SDP (4.2) with $\|y\|_1 \leq r$. Hence, using that $\mathscr{P}_0(X)$ is a *relaxation* of the feasible region of the dual (with upper bound ω on the objective value), we may restrict our advisor to return only such y:

$$\mathscr{P}_0(X) \neq \emptyset \Rightarrow \mathscr{P}_0(X) \cap \left\{ y \in \mathbb{R}^m : \sum_{j=1}^{m} y_j \leq r \right\} \neq \emptyset.$$

The benefit of this restriction is that an advisor that always returns a vector with bounded ℓ_1-norm automatically has a width $w^* \leq r+1$, due to the assumptions on the norms of the input matrices. The downside of this restriction is that the analogue of Lemma 4.2 does not hold for $\mathscr{P}_0(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$. However, we can find a primal solution using the SDP primal oracle we will present in Section 4.2.3.[10]

The following shows that an advisor that always returns a vector $y \in \tilde{\mathscr{P}}(\tilde{a}, c')$ if one exists, is a $4Rr\theta$-approximate advisor as in Definition 4.3.

Lemma 4.6 *Let $\tilde{a}_1, \ldots, \tilde{a}_m$ and $\tilde{c}$ be θ-approximations of $\operatorname{Tr}(A_1\rho), \ldots, \operatorname{Tr}(A_m\rho)$ and $\operatorname{Tr}(C\rho)$, respectively, where $X = R\rho$. Then the following holds:*

$$\mathscr{P}_0(X) \cap \left\{ y \in \mathbb{R}^m : \sum_{j=1}^{m} y_j \leq r \right\} \subseteq \tilde{\mathscr{P}}(\tilde{a}, c') \subseteq \mathscr{P}_{4Rr\theta}(X).$$

Proof. First, suppose $y \in \mathscr{P}_0(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$. We then have $y \in \tilde{\mathscr{P}}(\tilde{a}, c')$ because

$$\sum_{j=1}^{m} \tilde{a}_j y_j - \tilde{c} \geq \sum_{j=1}^{m} (\tilde{a}_j - \operatorname{Tr}(A_j\rho)) y_j - (\tilde{c} - \operatorname{Tr}(C\rho)) \geq -\theta \left\| y \right\|_1 - \theta \geq -(r+1)\theta,$$

where we first subtracted $\sum_{j=1}^{m} \operatorname{Tr}(A_j\rho) y_j - \operatorname{Tr}(C\rho) \geq 0$, and then used the triangle inequality.

Next, suppose $y \in \tilde{\mathscr{P}}(\tilde{a}, c')$. We show that $y \in \mathscr{P}_{4Rr\theta}(X)$. Indeed, since $|\operatorname{Tr}(A_j\rho) - \tilde{a}_j| \leq \theta$ we have

$$\operatorname{Tr}\left(\left(\sum_{j=1}^{m} y_j A_j - C \right) \rho \right) \geq \left(\sum_{j=1}^{m} \tilde{a}_j y_j + \tilde{c} \right) - (r+1)\theta \geq -(2 + r + \left\| y \right\|_1)\theta \geq -4r\theta,$$

where the last inequality used our assumptions $r \geq 1$ and $\left\| y \right\|_1 \leq r$. Hence

$$\operatorname{Tr}\left(\left(\sum_{j=1}^{m} y_j A_j - C \right) X \right) \geq -4r \operatorname{Tr}(X)\theta = -4Rr\theta.$$

For the latter inequality we use $\operatorname{Tr}(X) = R$. □

We have now seen the Arora-Kale framework for solving SDPs. To obtain a quantum SDP-solver it remains to implement a quantum advisor subroutine and supply it with good approximations of $\operatorname{Tr}(A\rho)$ values. By the above discussion it suffices to set $\theta = \varepsilon/(12Rr)$ and to use an advisor that is based on θ-approximations of $\operatorname{Tr}(A\rho)$ (for $A \in \{A_1, \ldots, A_m, C\}$), since with that choice of θ we have $\mathscr{P}_{4Rr\theta}(X) = \mathscr{P}_{\varepsilon/3}(X)$.

[10] Using several transformations of the SDP, from Section 4.7 and Lemma 2 of [BS17], one can show that there is a way to remove the need for this restriction. Hence, the Arora-Kale framework can also be used to find an approximate primal solution directly.

4.2.2 | An efficient 2-sparse advisor

In this section we will assume access to approximations $\tilde{a}_j$ of $\mathrm{Tr}(A_j\rho)$ and to $c' = \tilde{c} - r\theta - \theta$ as introduced in the last section. We will use these approximations to implement an advisor that always return a 2-sparse vector y. We will start with the strong assumption that we have a standard binary oracle for these approximations that acts as $|j\rangle|0\rangle|0\rangle \mapsto |j\rangle|\tilde{a}_j\rangle|\psi_j\rangle$, where $|\psi_j\rangle$ is some workspace state depending on j.

At the end of this section we briefly discuss how to modify the analysis when we have access to a relational quantum oracle that acts as $|j\rangle|0\rangle|0\rangle \mapsto |j\rangle \sum_i \beta_j^i \left|\tilde{a}_j^i\right\rangle \left|\psi_j^i\right\rangle$ where each $\left|\tilde{a}_j^i\right\rangle$ is an approximation of a_j and the amplitudes β_j^i are such that measuring the second register returns an $\tilde{a}_j^i$ which is θ-close to a_j with very high probability. Since our estimates will eventually come from a quantum algorithm, this more complicated form is often quite natural.

Our goal is to find a $y \in \tilde{\mathscr{P}}(\tilde{a}, c')$, i.e., a y such that

$$\begin{aligned} \|y\|_1 &\leq r \\ b^T y &\leq \omega \\ \tilde{a}^T y &\geq c' \\ y &\geq 0. \end{aligned}$$

Our first observation is that the polytope $\tilde{\mathscr{P}}(\tilde{a}, c')$ is extremely simple: it has only three non-trivial constraints and, if it is non-empty, then it contains a point with at most 2 non-zero coordinates (as we will show in Lemma 4.7).

A first naive approach to find a $y \in \tilde{\mathscr{P}}(\tilde{a}, c')$ would then be to find all vertices of the polytope, by solving $\Theta(m^2)$ linear systems of size 2 × 2 (this also determines if $\tilde{\mathscr{P}}(\tilde{a}, c')$ is non-empty). Here each linear system is determined by the values $\tilde{a}_j$, b_j and c', and thus we can decide if $\tilde{\mathscr{P}}(\tilde{a}, c')$ is non-empty with $\Theta(m^2)$ queries to these values.

We use a more sophisticated approach to show that $\Theta(m)$ classical queries suffice. Our approach is amenable to a quantum speedup: we show that only $\Theta(\sqrt{m})$ quantum queries suffice. In particular, we now show how to reduce the problem of finding a $y \in \tilde{\mathscr{P}}(\tilde{a}, c')$ to finding a convex combination of points $(b_j, \tilde{a}_j)$ that lies within a certain region of the plane.

First observe that if $\omega \geq 0$ and $c' \leq 0$, then $y = 0$ is a solution and our advisor can return it. From now on we will assume that $\omega < 0$ or $c' > 0$. Then for a feasible point y we may write $y = Nq$ with $N = \|y\|_1 > 0$ and hence $\|q\|_1 = 1$. So we are looking for an N and a q such that

$$\begin{aligned} b^T q &\leq \omega/N \\ \tilde{a}^T q &\geq c'/N \\ \|q\|_1 &= 1 \\ q &\geq 0 \\ 0 < N &\leq r. \end{aligned} \tag{4.8}$$

We can now view $q \in \mathbb{R}^m_{>0}$ as the coefficients of a convex combination of the points $p_i := (b_i, \tilde{a}_i)$ in the plane. We want such a combination that lies to the upper left of $g_N := (\omega/N, c'/N)$ for some $0 < N \leq r$. Let $\mathcal{G}_N$ denote the upper-left quadrant of the plane starting at g_N.

Lemma 4.7 *If there is a $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$, then there is a 2-sparse $y' \in \tilde{\mathcal{P}}(\tilde{a}, c')$ such that $\|y\|_1 = \|y'\|_1$.*

Proof. Consider $p_i = (b_i, \tilde{a}_i)$ and $g = (\omega/N, c'/N)$ as before, and write $y = Nq$ where $\sum_{j=1}^m q_j = 1$, $q \geq 0$. The vector q certifies that a convex combination of the points p_i lies in $\mathcal{G}_N$. But then there exist $j, k \in [m]$ such that the line segment $\overline{p_j p_k}$ intersects $\mathcal{G}_N$. All points on this line segment are convex combinations of p_j and p_k, hence there is a convex combination of p_j and p_k that lies in $\mathcal{G}_N$. This gives a 2-sparse q', and $y' = Nq' \in \tilde{\mathcal{P}}(\tilde{a}, c')$. □

We can now restrict our search to 2-sparse y. Let $\mathcal{G} := \bigcup_{N \in (0,r]} \mathcal{G}_N$, see Figure 4.1 for the shape of $\mathcal{G}$. Then we want to find two points p_j, p_k that have a convex combination in $\mathcal{G}$, since this implies that a scaled version of their convex combination gives a $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$ with $\|y\|_1 \leq r$ (this scaling can be computed efficiently given p_j and p_k).

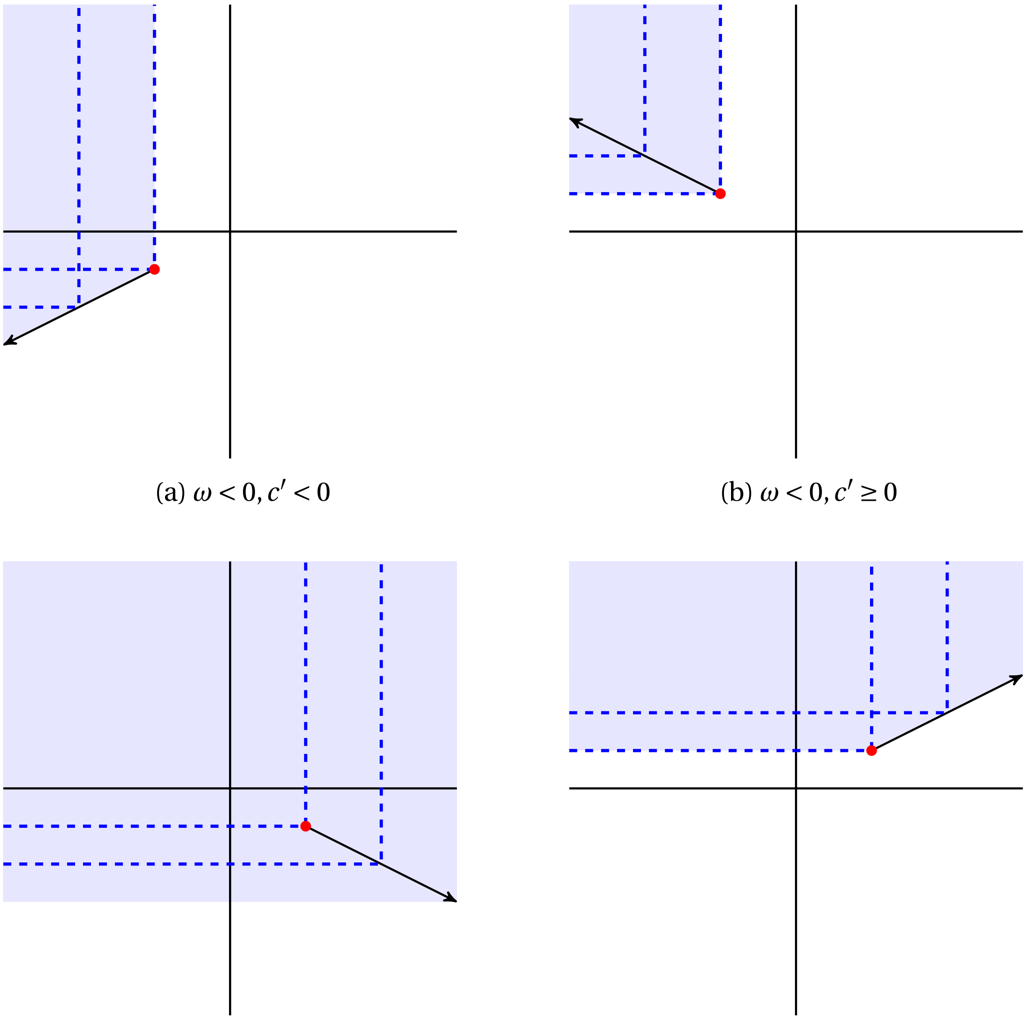
Furthermore, regarding the possible (non-)emptiness of $\mathcal{G}$ we know the following by Lemma 4.6 and Lemma 4.7:

- If $\mathcal{P}_0(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$ is non-empty, then some convex combination of two of the p_j's lies in $\mathcal{G}$.
- If $\mathcal{P}_{4Rr\theta}(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$ is empty, then no convex combination of the p_j's lies in $\mathcal{G}$.

Lemma 4.8 *There is an algorithm that returns a 2-sparse vector q such that $\sum_{j=1}^m q_j p_j \in \mathcal{G}$, if one exists, using one search and two minimizations over the m points $p_j = (b_j, \tilde{a}_j)$. This gives a classical algorithm that uses $\mathcal{O}(m)$ calls to the subroutine that gives the entries of $\tilde{a}$, and $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(m)$ other operations; and a bounded-error quantum algorithm that uses $\mathcal{O}(\sqrt{m})$ calls to an (exact quantum) subroutine that gives the entries of $\tilde{a}$, and $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\sqrt{m})$ elementary operations.*

Proof. The algorithm can be summarized as follows:

1. Check if $\omega \geq 0$ and $c' \leq 0$. If so, then return $q = 0$.
2. Check if there is a $p_i \in \mathcal{G}$. If so, then return $q = e_i$
3. Find p_j, p_k so that the line segment $\overline{p_j p_k}$ goes through $\mathcal{G}$ and return the corresponding q.
4. If the first three steps did not return a vector q, then output ‘Fail’.



(a) $\omega < 0, c' < 0$

(b) $\omega < 0, c' \geq 0$

(c) $\omega \geq 0, c' < 0$

(d) $\omega \geq 0, c' \geq 0$

Figure 4.1: The region $\mathscr{G}$ in light blue. The borders of two quadrants $\mathscr{G}_N$ have been drawn by thick dashed blue lines. The red dot at the beginning of the arrow is the point $(\omega/r, c'/r)$.

The main realization is that in step 3 we can search separately for p_j and p_k. We explain this in more detail below, but first we will need a better understanding of the shape of $\mathcal{G}$ (see Figure 4.1 for illustration). The shape of $\mathcal{G}$ depends on the sign of ω and c'.

(a) If $\omega < 0$ and $c' < 0$. The corner point of $\mathcal{G}$ is $(\omega/r, c'/r)$. One edge goes up vertically and an other follows the line segment $\lambda \cdot (\omega, c')$ for $\lambda \in [1/r, \infty)$ starting at the corner.

(b) If $\omega < 0$ and $c' \geq 0$. Here $\mathcal{G}_N \subseteq \mathcal{G}_r$ for $N \leq r$. So $\mathcal{G} = \mathcal{G}_r$. The corner point is again $(\omega/r, c'/r)$, but now one edge goes up vertically and one goes to the left horizontally.

(c) If $\omega \geq 0$ and $c' \leq 0$. This is the case where $y = 0$ is a solution, $\mathcal{G}$ is the whole plane and has no corner.

(d) If $\omega \geq 0$ and $c' > 0$. The corner point of $\mathcal{G}$ is again $(\omega/r, c'/r)$. From there one edge goes to the left horizontally and one edge follows the line segment $\lambda \cdot (\omega, c')$ for $\lambda \in [1/r, \infty)$.

Since $\mathcal{G}$ is always an intersection of at most 2 halfspaces, steps 1-2 of the algorithm are easy to perform. In step 1 we handle case (c) by simply returning $y = 0$. For the other cases $(\omega/r, c'/r)$ is the corner point of $\mathcal{G}$ and the two edges are simple lines. Hence in step 2 we can easily search through all the points to find out if there is one lying in $\mathcal{G}$; since $\mathcal{G}$ is a very simple region, this only amounts to checking on which side of the two lines a point lies.

Now, if we cannot find a single point in $\mathcal{G}$ in step 2, then we need a combination of two points in step 3. Let L_1, L_2 be the edges of $\mathcal{G}$ and let ℓ_j and ℓ_k be the line segments from $(\omega/r, c'/r)$ to p_j and p_k, respectively. Then, as can be seen in Figure 4.2, the line segment $\overline{p_j p_k}$ goes through $\mathcal{G}$ if and only if (up to relabeling p_j and p_k) $\angle \ell_j L_1 + \angle L_1 L_2 + \angle L_2 \ell_k \leq \pi$. Since $\angle L_1 L_2$ is fixed, we can simply look for a j such that $\angle \ell_j L_1$ is minimized and a k such that $\angle L_2 \ell_k$ is minimized. If $\overline{p_j p_k}$ does not pass through $\mathcal{G}$ for this pair of points, then it does not for any of the pairs of points.

Notice that these minimizations can be done separately and hence can be done in the stated complexity (using quantum minimum-finding [DH96] for the quantum algorithm). Given the minimizing points p_j and p_k, it is easy to check if they give a solution by calculating the angle between ℓ_j and ℓ_k. The coefficients of the convex combination q are then easy to compute. □

The analysis above applies if there are m points $p_j = (b_j, \tilde{a}_j)$, where $j \in [m]$, and we are given a unitary which acts as $|j\rangle|0\rangle|0\rangle \mapsto |j\rangle|\tilde{a}_j\rangle|\psi_j\rangle$. We now consider the more general case where we are given access to a unitary which for each j provides a superposition over different values $\tilde{a}_j$. That is, we assume that we are given an oracle that acts as $|j\rangle|0\rangle|0\rangle \mapsto |j\rangle \sum_i \beta_j^i \left|\tilde{a}_j^i\right\rangle \left|\psi_j^i\right\rangle$ where each $\left|\tilde{a}_j^i\right\rangle$ is an approximation of a_j and

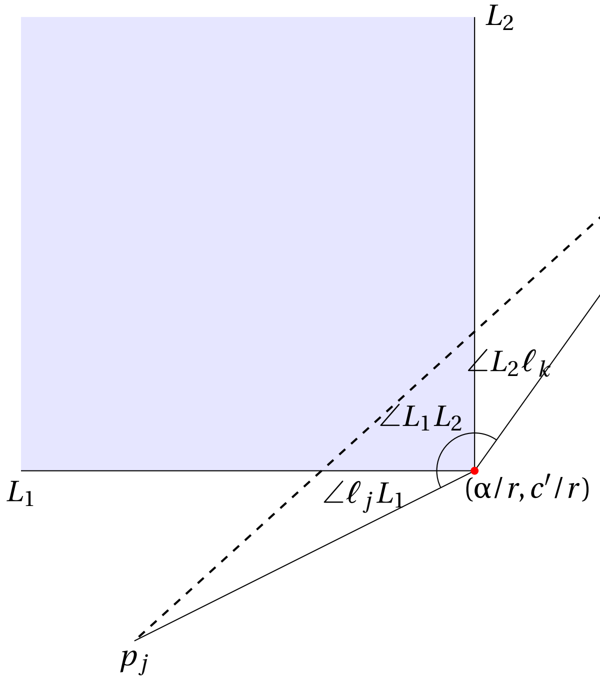


Figure 4.2: Illustration of $\mathcal{G}$ with the points p_j, p_k and the angles $\angle\ell_jL_1, \angle L_1L_2, \angle L_2\ell_k$ drawn in. Clearly the line $\overline{p_jp_k}$ only crosses $\mathcal{G}$ when the total angle is less than π.

the amplitudes β^i_j are such that measuring the second register with high probability returns an $\tilde{a}^i_j$ which is θ-close to a_j. We do so because the procedure to estimate the trace that we will implement later provides an oracle of this form.

Since we can exponentially reduce the probability that we obtain an $\tilde{a}^i_j$ which is further than θ away from a_j, we will for simplicity assume that for all i, j we have $|\tilde{a}^i_j - a_j| \le \theta$; the neglected exponentially small probabilities will only affect the analysis in negligible ways. We will need the following lemma:

Lemma 4.9 : Generalized Minimum-Finding [AGGW17, Thm. 49] *Let U be a unitary, acting on q qubits, such that $U|0\rangle = \sum_{k=1}^N \alpha_k|\psi_k\rangle|x_k\rangle$, where the x_k are real numbers represented in binary and the $|\psi_k\rangle$ are arbitrary quantum states. Let $x_1 < x_2 < \ldots < x_N$ and define X to be the random variable obtained by measuring the second register, i.e., $\Pr[X = x_k] = |\alpha_k|^2$. Let $x \in \mathbb{R}$ and $p \in \mathbb{R}_{>0}$ such that $p \le \Pr[X < x]$. Then there is an algorithm that outputs an x_i such that $x_i \le x$ with probability at least $1-\delta$ and uses $\mathcal{O}(\log(1/\delta)/\sqrt{p})$ applications of U and $U^\dagger$ and $\mathcal{O}(\log(1/\delta)q/\sqrt{p})$ elementary operations.*

Let $p^i_j := (b_j, \tilde{a}^i_j)$. Our new goal will be to find a 2-sparse vector q and points p^i_j and $p^{i'}_k$ such that $q_jp^i_j + q_kp^{i'}_k \in \mathcal{G}$, or to conclude that for all $j, k \in [m]$ there exist i and i' such that no q exists for which $q_jp^i_j + q_kp^{i'}_k \in \mathcal{G}$.

Note that while we do not allow our quantum algorithm enough time to obtain classical descriptions of all $\tilde{a}_j$s (we aim for a runtime of $\widetilde{\mathcal{O}}(\sqrt{m})$), we do have enough time to compute $\tilde{c}$ once initially (after this measurement $\mathcal{G}$ is well-defined). Knowing $\tilde{c}$, we can compute the angles defined by the points $p_j^i = (b_j, \tilde{a}_j^i)$ with respect to the corner point of $(\omega/r, (\tilde{c}-\theta)/r-\theta)$ and the lines L_1, L_2 (see Figure 4.2).

We now apply the Generalized Minimum-Finding algorithm, where U is the unitary that sets up a uniform superposition over the js, queries the oracle for $\tilde{a}_j$, and calculates the relevant angles from the result. With $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}\left(\sqrt{m}\right)$ queries and elementary operations we find $k, \ell \in [m]$ and points p_k^i and $p_\ell^{i'}$ approximately minimizing the respective angles to lines L_1, L_2. Here 'approximately minimizing' means that there is no $j \in [m]$ such that for all i'' the angle of $p_j^{i''} = (b_j, \tilde{a}_j^{i''})$ with L_1 is smaller than that of p_k^i with L_1 (and similarly for ℓ and L_2). From this point on we can simply consider the model in Lemma 4.8 since by the analysis above there exists an approximation $\tilde{a} \in \mathbb{R}^m$ with $\tilde{a}_k = \tilde{a}_k^i$ and $\tilde{a}_\ell = \tilde{a}_\ell^{i'}$ and where k and ℓ are the correct minimizers.

We want to stress that our solver is meant to work for all SDPs. In particular, our advisor does not use the structure of a specific SDP. This might lead to an unnecessary large width-bound; to obtain quantum speedups for a *specific* class of SDPs it might be necessary to develop advisors tuned to that problem. We view this as an important direction for future work. Recall from the introduction that Arora and Kale also obtain fast classical algorithms for problems such as MAXCUT by developing specialized (classical) advisors.

4.2.3 | An SDP primal oracle

As discussed in Section 4.2.1, the Arora-Kale algorithm returns a primal solution if at some stage the approximate advisor cannot return a solution. However, due to the additional constraint $\|y\| \leq r$ that we add to the polytope for the advisor, this is no longer the case. In this section we will construct an SDP primal oracle in order to obtain descriptions of primal solutions as well.

To construct an SDP primal oracle, we use the same algorithm as Brandão et al. [BKLLSW19], which comes from the proof of Lemma 4.6 in the work of Lee, Raghavendra and Steurer [LRS15]. Since we will mostly focus on the Arora-Kale framework we will only give a short sketch here.

The algorithm we will use only solves feasibility problems with an equality constraint $\mathrm{Tr}(X) = R$ for the trace, hence a few small reductions are required to apply this technique. First we will add the objective as a constraint by letting $A_0 = -C$ and $b_0 = -\omega$. Then, we add one new variable denoted by w such that

$$X' := \begin{bmatrix} X & 0 \\ 0 & w \end{bmatrix}.$$

Now $\mathrm{Tr}\left(X'\right) = R$ and $X \succeq 0$ imply that $\mathrm{Tr}(X) \leq R$, and we get a new SDP that is equivalent to the previous one. It can be shown that in our input models this reduction does not introduce more than a constant factor overhead in the complexity. We can now apply Meta-Algorithm 4.2 with precision ε to construct an ε-approximate SDP primal oracle.

Input Hermitian matrices $A_0, \dots, A_m \in \mathbb{R}^{n\times n}$ and reals $b_0, \dots, b_m \in [-R, R]$. An additive error parameter $\varepsilon > 0$.

Output Either one of the following:

- A vector $y \in \mathbb{R}^{m+1}_{\geq 0}$ such that for $X = Re^{-\sum_{j=0}^m y_j^{(t-1)} A_j} / \mathrm{Tr}\left(e^{-\sum_{j=0}^m y_j^{(t-1)} A_j}\right)$ we have $\mathrm{Tr}\left(A_j X\right) \leq b_j + \varepsilon$ for $j = 0, \dots, m$.
- "Infeasible", indicating that no psd X exists with $\mathrm{Tr}(X) = R$ such that $\mathrm{Tr}\left(A_j X\right) \leq b_j$ for $j = 0, \dots, m$.

```
$y^{(0)} := 0 \in \mathbb{R}^{m+1}$.
$\theta = \frac{\varepsilon}{2R}$.
$T := \frac{\ln(n)}{\theta^2}$.
for $t = 1, \dots, T$ do
    $\rho^{(t)} := \frac{e^{-\sum_{j=0}^m y_j^{(t-1)} A_j}}{\mathrm{Tr}\left(e^{-\sum_{j=0}^m y_j^{(t-1)} A_j}\right)}$.
    if $\exists j$ s.t. $\mathrm{Tr}\left(A_j \rho^{(t)}\right) \geq b_j / R$ then
        Let $j$ be one such index.
        $y^{(t)} := y^{(t-1)} + \theta e_j$.
    alternatively if $\forall j : \mathrm{Tr}\left(A_j \rho^{(t)}\right) \leq b_j / R + \theta$ then
        Return $y^{(t)}$.
    end if
end for
Output "Infeasible".
```

Meta-Algorithm 4.2: The SDP primal oracle.

Theorem 4.10 : [LRS15, Proof of Lemma 4.6] *Suppose we are given input matrices $A_0, \dots, A_m$ that all have operator norm at most 1, and input reals $b_0, \dots, b_m$ that are all in $[-R, R]$. If Meta-Algorithm 4.2 outputs vector y, then the matrix*

$$X := R \frac{e^{-\sum_{j=0}^m y_j A_j}}{\mathrm{Tr}\left(e^{-\sum_{j=0}^m y_j A_j}\right)}$$

is such that $\mathrm{Tr}\left(X A_j\right) \leq b_j + \varepsilon$ for all j. If Meta-Algorithm 4.2 outputs "infeasible" then there is no matrix X with trace exactly R that satisfies $\mathrm{Tr}\left(A_j X\right) \leq b_j$ for all j.

Both frameworks have a very similar structure. The main difference is that the SDP primal oracle framework (Meta-Algorithm 4.2) requires only a simple search, whereas the θ-approximate advisor needed for the Arora-Kale framework is slightly more complex. Furthermore the SDP primal oracle has $\gamma = R/\varepsilon$ as a relevant parameter, where the Arora-Kale framework considered the parameter $\gamma = Rr/\varepsilon$. However, as discussed at the end of Section 4.1.1, to use an SDP primal oracle for finding the optimal value of an SDP up to error ε we will need to scale down the error of the SDP primal oracle by a factor r, leading to a similar γ.

The implementation of the advisor that we use for the Arora-Kale framework always returns a 2-sparse vector, thus in both cases we will work with a y vector that is non-negative, that will be $\widetilde{\mathcal{O}}_n(\gamma^2)$-sparse, and that has ℓ_1-norm at most $\widetilde{\mathcal{O}}_n(\gamma)$ (for the relevant choice of γ).

4.2.4 | Subroutines

There are two major subroutines which we will need to implement for both meta-frameworks. First, the algorithms will require an implementation of a Gibbs-sampler.

Definition 4.11 : Gibbs-sampler *A θ-precise* Gibbs-sampler *for the input matrices $A_0,\dots,A_m$ is a unitary that takes as input a data structure storing a vector $y\in\mathbb{R}^{m+1}_{\geq 0}$ and creates as output a purification of a θ-approximation in trace distance of the Gibbs state $e^{-\sum_{j=0}^m y_jA_j}/\mathrm{Tr}\left(e^{-\sum_{j=0}^m y_jA_j}\right)$. If $\|y\|_1\leq K$ and the support of y has size at most d, then we write $T_{Gibbs}(K,d,\theta)$ for the optimal number of input queries needed to Gibbs sample.*

Note that in physics a Gibbs-sampler is often a procedure that prepares the Gibbs state as a *mixed state*, our definition is more demanding. We will use the approximate Gibbs states in order to estimate the quantity $\mathrm{Tr}(A_j\rho)$.

Definition 4.12 : Trace-estimator *A (θ,σ)*-trace-estimator *is a unitary that as input takes an index j and acts on a state ρ, such that measuring part of the resulting state gives a sample from a random variable $X_j\in\mathbb{R}$. Here X_j is an estimate of the trace that is at most $\frac{\theta}{4}$-biased:*

$$|\mathrm{Tr}(A_j\rho)-\mathbb{E}[X_j]|\leq\theta/4,$$

and the standard deviation of X_j is at most σ. We write $T^{\sigma}_{Tr}(\theta)$ for the optimal number of queries to an input oracle for A_j that such a procedure needs to make.

If given access to both of these subroutines, then we can construct an SDP-solver, as we will prove below. Before we state the theorem we note that in both meta-algorithms $\theta=\mathcal{O}(\gamma^{-1})$, and that in the Arora-Kale framework $\eta=\widetilde{\mathcal{O}}_n(\theta)$.

Theorem 4.13 *Assume that $y^{(t)}\in\mathbb{R}^{m+1}$ is stored in a data structure such that updating an entry requires at most $\widetilde{\mathcal{O}}(T_{Gibbs}(\gamma,\gamma^2,\gamma^{-1}))$ elementary operation, where $\gamma:=6Rr/\varepsilon$. Furthermore, assume that both $T_{Gibbs}(\gamma,\gamma^2,\gamma^{-1})$ and $T^{\sigma}_{Tr}(\gamma^{-1})$ are at most polynomial in their parameters. Then*

$$T_{SDP}(\varepsilon)=\widetilde{\mathcal{O}}_n\left(\sqrt{m}\left(T^{\sigma}_{Tr}(\gamma^{-1})+T_{Gibbs}(\gamma,\gamma^2,\gamma^{-1})\right)\gamma^3\sigma\right).$$

Similarly there is also a quantum algorithm with the same complexity, but with $\gamma:=6R/\varepsilon$, that implements an SDP primal oracle.

Proof. To obtain an ε-approximate SDP-solver we use binary search and the Arora-Kale framework to find an $\varepsilon/4$-approximation $\widetilde{\mathrm{OPT}}$ of OPT. Then we run the Arora-Kale algorithm with guess $\omega = \widetilde{\mathrm{OPT}} + \frac{3}{4}\varepsilon$ and precision $\varepsilon/4$ to find a feasible dual solution that is ε-optimal. Finally, we run the SDP primal oracle with $\omega = \widetilde{\mathrm{OPT}} - \frac{3}{4}\varepsilon$ and precision $\varepsilon/4$ in order to obtain a primal solution that is ε-feasible and has an objective value that is ε-close to optimal.

Since both frameworks consist of $\widetilde{\mathcal{O}}_n(\gamma^2)$ iterations, we know that the $y^{(t)}$ vector will be at most $\widetilde{\mathcal{O}}_n(\gamma^2)$-sparse during the run of the algorithm and will have ℓ_1-norm at most $\widetilde{\mathcal{O}}_n(\gamma)$. This allows us to prepare a purification of ρ that is $\frac{\theta}{8}$-close in trace distance using $\widetilde{\mathcal{O}}_n\left(T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\right)$ queries. By applying the trace-estimator with precision $\theta/8$ we can construct a random variable that, when measured, outputs a number that is $\theta/8 + \theta/8 = \theta/4$-close in expectation to $\mathrm{Tr}(A_j\rho)$.

A relational oracle for θ-approximations of the quantities $\mathrm{Tr}(A_j\rho)$ can be constructed by using amplitude estimation techniques [Mon15, Thm. 5] with precision $\theta/2$ on this random variable using

$$\widetilde{\mathcal{O}}_n\left(\left(T^{\sigma}_{Tr}(\gamma^{-1}) + T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\right)\gamma\sigma\right)$$

queries in total.

Using amplitude amplification to find a violated constraint, or using generalized minimum-finding to minimize the angles within the advisor for the Arora-Kale framework, allows us to implement each iteration using

$$\widetilde{\mathcal{O}}_n\left(\sqrt{m}\left(T^{\sigma}_{Tr}(\gamma^{-1}) + T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\right)\gamma\sigma\right)$$

queries. Summing this over all $T = \widetilde{\mathcal{O}}_n(\gamma^2)$ iterations gives the stated bound. □

4.3 | Gibbs-sampling and trace-estimators

We are going to work with Hermitian matrices $A, H \in \mathbb{C}^{n\times n}$, and write ρ for the Gibbs state $e^H/\mathrm{Tr}(e^H)$. Our goal will be to prepare the state ρ and implement a trace-estimator for A. We will first consider the setting in which we have access to block-encodings of A and H. We will then show how to implement such a block-encoding when given sparse access to A and H. We will discuss how to obtain query access to H (either as a sparse matrix or as a block-encoding) in later sections.

4.3.1 | Gibbs-sampling and trace-estimators using block-encodings

In this section we show how to implement a Gibbs-sampler and a trace-estimator when we are given block-encodings of A and H. We will use the following polynomial approximation of the exponential function:

Lemma 4.14 *Let $\xi \in (0, 1/6]$ and $\beta \geq 1$. There exists a polynomial $P(x)$ such that*

- *For all* $x \in [-1,0]$ *we have* $\left|P(x) - e^{2\beta x}/4\right| \leq \xi$.
- *For all* $x \in [-1,1]$ *we have* $|P(x)| \leq 1/2$.
- $\deg(P) = \widetilde{\mathcal{O}}_{\frac{1}{\xi}}(\beta)$.

Proof. This is shown in [GSLW19, Cor. 64]. It can also be proven with an argument similar to the example in Section 2.3.4. □

We can now use this approximation lemma to prove our Gibbs-sampling lemma.

Lemma 4.15 *Let* $\theta \in (0,1/3]$, $\beta > 1$, *and let* d *be the degree of the polynomial from Lemma 4.14 when we let* $\xi = \frac{\theta}{128n}$. *Let* U *be a* $\left(\beta, a, \frac{\theta^2\beta}{1024^2 d^2 n^2}\right)$*-block-encoding*[11] *of a Hermitian operator* $H \in \mathbb{R}^{n\times n}$, *i.e., a* $\left(\beta, a, \widetilde{\mathcal{O}}(\theta^2/\beta n^2)\right)$*-block-encoding. Then we can create purification of a state* $\tilde{\rho}$ *such that*

$$\left\|\tilde{\rho} - \frac{e^H}{\mathrm{Tr}\left(e^H\right)}\right\|_{tr} \leq \theta$$

using $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\sqrt{n}\beta)$ *applications of* U *and* $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\sqrt{n}\beta a)$ *elementary operations.*

Proof. Consider the spectral decomposition of H

$$H = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle\langle\psi_i|.$$

We would like to prepare a purification of the state

$$\rho = \frac{1}{\sum_{i=1}^n e^{\lambda_i}} \sum_{i=1}^{n} e^{\lambda_i} |\psi_i\rangle\langle\psi_i|.$$

We will use the fact that a maximally entangled state is basis independent:

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle|i\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}|\psi_i\rangle|\bar{\psi}_i\rangle$$

for all orthogonal bases $\{|\psi_i\rangle\}$, where $\bar{\psi}_i$ is equal to the state $|\psi_i\rangle$ but with all amplitudes conjugated.

The idea is to transform the block-encoding of H by the function $f(x) = e^{\beta x/2}$, apply the new block-encoding to a uniform superposition of the eigenvectors of H (by applying it to one half of the maximally entangled state), and then use amplitude amplification to obtain the correct state. However, to apply $f(x) = e^{\beta x/2}$ we need that the λ_i are non-positive in order to ensure that $\|f(H)\| \leq 1$. Furthermore, to apply amplitude amplification efficiently we need that at least one of the $e^{\lambda_i/2}$ is relatively large.

[11] A more careful error analysis can drastically reduce the constants here. In our applications we will always be able to reduce the error in the block-encoding of H exponentially.

To achieve both of these requirements we will shift the spectrum of H. Note that this does not influence the final state:

$$e^{H+\lambda I}/\mathrm{Tr}\left(e^{H+\lambda I}\right) = e^{H}/\mathrm{Tr}\left(e^{H}\right) \text{ for all } \lambda \in \mathbb{R}.$$

We would like to shift by $-\lambda_{\max}(H)$ in order to make all the eigenvalues non-positive, and hence first need to approximate this value. We can do so by using the Generalized Minimum-Finding algorithm (Lemma 4.9) in a similar way as Lemma 50 from [AGGW17] does for the sparse matrix input model. The proof of this Lemma 50 only uses the input to implement Hamiltonian simulation under H, which we can achieve using Theorem 2.10. Roughly the procedure is as follows (see [AGGW17, Lem. 50] for details):

1. Set up a maximally entangled state
$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle|i\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}|\psi_i\rangle|\bar{\psi}_i\rangle.$$

2. Apply phase estimation with precision $\frac{1}{5\beta}$ with $e^{-\mathbf{i}H/\beta}$ applied to the left half of the maximally entangled state to obtain[12]
$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|\psi_i\rangle|\bar{\psi}_i\rangle\sum_{j=1}^{K}\beta_j\left|-\tilde{\lambda}_i^{(j)}\right\rangle.$$
We do this in such a manner that measuring the last register results in a $\tilde{\lambda}_i^{(j)}$ with $\left|\lambda_i - \tilde{\lambda}_i^{(j)}\right| \leq 1/4$ with probability at least $1-\mu$, for some very small μ.

3. Use the Generalized Minimum-Finding algorithm to find a $\tilde{\lambda}_i^{(k)}$ that is $\frac{1}{4}$-close to $\lambda_{\max}(H)$.

This procedure uses $\widetilde{\mathcal{O}}\left(\sqrt{n}/\beta\right)$ queries to a block-encoding of H. Let $\tilde{\lambda}_{\max}$ be our approximation of the largest eigenvalue. Since we can efficiently reduce the error probability (exponentially in the number of repetitions, and by decreasing μ) we will for the rest of the proof assume this procedure was successful.

Now we can use the linear combination of block-encodings lemma (Lemma 2.9) to implement a $\left(4\beta, \mathcal{O}(a), \frac{\theta^2\beta}{512^2 d^2 n^2}\right)$-block-encoding of $H' = H - (\tilde{\lambda}_{\max} + 1/4)I$. Let λ'_i denote the eigenvalues of H'. We now know that $\frac{\lambda'_i}{4\beta} \in [-1, 0]$ and that $-1/2 \leq \lambda_{\max}(H') \leq 0$. Note that

$$\frac{e^{H'}}{\mathrm{Tr}\left(e^{H'}\right)} = \frac{e^{H}}{\mathrm{Tr}\left(e^{H}\right)}.$$

[12] Here we need a slightly higher precision in the phase estimation than the error we want to achieve on λ_i, due to the fact that our block-encoding of H is not exact.

We can use Lemma 2.11 and Lemma 4.14 to obtain a $(1,\mathcal{O}(a),\frac{\theta}{64n})$-block-encoding of $e^{H'/2}/4$ using $\widetilde{\mathcal{O}}_{\frac{n}{\theta}}(\beta)$ applications of U and $\widetilde{\mathcal{O}}_{\frac{n}{\theta}}(a\beta)$ elementary operations. We apply this block-encoding to the first register of the maximally entangled state to get a state that is $\frac{\theta}{64n}$-close in ℓ_2-norm to

$$|\phi\rangle := \frac{1}{\sqrt{n}}\sum_{i=1}^{n} e^{\lambda_i'/2}/4|\psi_i\rangle|\bar{\psi}_i\rangle|0\rangle + |\perp\rangle,$$

where the last register consists of $\mathcal{O}(a)$ qubits and $|\perp\rangle$ is an arbitrary state that does not have any overlap with the all-zero state in the last register. Tracing out the second register of $\frac{1}{\sqrt{n}}\sum_{i=1}^{n} e^{\lambda_i'/2}/4|\psi_i\rangle|\bar{\psi}_i\rangle$ gives the sub-normalized density matrix that is $\frac{\theta}{64n}$-close in trace norm to

$$\frac{1}{16n}\sum_{i=1}^{n} e^{\lambda_i'}|\psi_i\rangle\langle\psi_i|.$$

Since at least one of the λ_i' is larger than $-1/2$ we know that the probability of measuring a zero in the last register of $|\phi\rangle$ is at least (ignoring the small $\theta/64n$ error)

$$\mathrm{Tr}\left(\frac{1}{16n}\sum_{i=1}^{n} e^{\lambda_i'}|\psi_i\rangle\langle\psi_i|\right) \geq \frac{e^{-1/2}}{16n} \geq \frac{1}{32n}.$$

It follows that $\mathcal{O}(\sqrt{n})$ rounds of amplitude amplification suffice to get a state that is θ-close in ℓ_2-norm to

$$\frac{1}{\sqrt{\sum_{i=1}^{n} e^{\lambda_i'/2}}}\sum_{i=1}^{n} e^{\lambda_i'/2}|\psi_i\rangle|\bar{\psi}_i\rangle|0\rangle.$$

Since this state is a purification of the Gibbs state that we wanted to prepare, it follows that $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\sqrt{n}\beta)$ uses of U and $U^\dagger$, and $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(a\beta\sqrt{n})$ elementary operations suffice. □

Now that we have a way of creating approximate purifications of ρ, we would like to construct a trace-estimator for $\mathrm{Tr}(A\rho)$. A first attempt could be to apply A as a block-encoding to the first register of our purification of ρ, and then use amplitude estimation to approximate how much the state gets subnormalized. However, applying A directly gives us a purification of the subnormalized density operator $A\rho A^\dagger$, which is subnormalized by a factor $\mathrm{Tr}(A^2\rho)$, not $\mathrm{Tr}(A\rho)$. Hence we instead would like to apply $\sqrt{A}$ to ρ. Since in our applications we will have a block-encoding where the A matrix is scaled down we will use the following approximation of the linear function $\alpha x/4$ (for x close to the origin) to scale it up.

Lemma 4.16 : [GSLW19, Proof of Lemma 30] *Let $\xi \in (0,1/2)$ and $\alpha \geq 4$. There exists a polynomial $R(x)$ such that*

- *For all $x \in [-1,1]$ we have $|R(x)| = |R(-x)| \leq 1$.*
- *For all $x \in [-1/\alpha, 1/\alpha]$ we have $|R(x) - \alpha x/4| \leq \xi$.*

- $\deg(R) = \mathcal{O}\big(\alpha \log(1/\xi)\big)$.

We now prove the following lemma that gives a polynomial approximation of the square root function.

Lemma 4.17 *Let $\theta \in (0, 1/6]$ and $\alpha \geq 4$. There exists a polynomial $Q(x)$ such that*

- *For all $x \in [-1/\alpha, 1/\alpha]$ we have $\left|Q(x) - \frac{1}{4}\sqrt{1 + \alpha x/8}\right| \leq \theta$.*
- *For all $x \in [-1, 1]$ we have $|Q(x)| \leq 1/2$.*
- $\deg(Q) = \widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha)$.

Proof. We first consider a polynomial approximation for the function $\frac{1}{4}\sqrt{1 + x/2}$. Consider the Taylor series of $\frac{1}{4}\sqrt{1 + x/2}$ around $x = 0$:

$$\frac{1}{4}\sum_{k=0}^{\infty} 2^{-k}\binom{1/2}{k} x^k,$$

which converges for all $x \in [-1, 1]$. Let $Q_0(x) = \frac{1}{4}\sum_{k=0}^{B} 2^{-k}\binom{1/2}{k} x^k$ for $B = \lceil \log(1/\theta) \rceil$. Then for all $x \in [-1, 1]$ we have that

$$\begin{aligned}
\left|Q_0(x) - \frac{1}{4}\sqrt{1 + x/2}\right| &= \left|\frac{1}{4}\sum_{k=B+1}^{\infty} \frac{1}{2^k}\binom{1/2}{k} x^k\right| \\
&\leq \frac{1}{4}\sum_{k=B+1}^{\infty} \left|\frac{1}{2^k}\binom{1/2}{k}\right| \left|x^k\right| \\
&\leq \frac{1}{4}\sum_{k=B+1}^{\infty} \left|\frac{1}{2^k}\binom{1/2}{k}\right| \\
&\leq \frac{1}{4}\sum_{k=B+1}^{\infty} \left(\frac{1}{2}\right)^k \\
&= \frac{1}{4}\cdot\frac{\left(\frac{1}{2}\right)^B}{1 - \frac{1}{2}} \\
&\leq \theta/2,
\end{aligned}$$

where we used that

$$\left|\binom{1/2}{k}\right| = \left|\frac{\frac{1}{2}\left(\frac{1}{2} - 1\right)\cdots\left(\frac{1}{2} - k + 1\right)}{k!}\right| \leq 1.$$

Since $\frac{1}{4}\sqrt{1 + x/2} \in [0, 1/3]$ for all $x \in [-1, 1]$ we know that $Q_0(x) \in [-1/2, 1/2]$ for all $x \in [-1, 1]$.

Now let $R(x)$ be the polynomial from Lemma 4.16 with precision $\xi = \theta/\alpha$. Let $Q(x) = Q_0(R(x))$. Then for $x \in [-1,1]$ we have $Q(x) \in [-1/2,1/2]$ since $R(x) \in [-1,1]$. Furthermore, for $x \in [-1/\alpha, 1/\alpha]$

$$\begin{aligned}\left|Q(x) - \frac{1}{4}\sqrt{1+\alpha x/8}\right| &\leq \frac{\theta}{2} + \left|\frac{1}{4}\sqrt{1+R(x)/2} - \frac{1}{4}\sqrt{1+\alpha x/8}\right| \\ &\leq \frac{\theta}{2} + \frac{\theta}{\alpha}\max_{y\in(-2/\alpha,2/\alpha)}\left|\frac{\alpha}{4\cdot 16\sqrt{1+\alpha y/8}}\right| \\ &\leq \frac{\theta}{2} + \frac{\theta}{\alpha}\left|\frac{\alpha}{32}\right| \\ &\leq \theta.\end{aligned}$$

□

We are now ready to implement a trace-estimator.

Lemma 4.18 *Let ρ be an n-dimensional quantum state and U be an $(\alpha, a, \theta/2)$-block-encoding of a matrix $A \in \mathbb{R}^{n\times n}$ with $\|A\| \leq 1$. A trace-estimator for $\mathrm{Tr}(A\rho)$ with bias at most θ and $\sigma = \mathcal{O}(1)$ can be implemented using $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha)$ uses of U and $U^\dagger$ and $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha)$ elementary operations. Hence, in the quantum operator input model*

$$T_{Tr}^{\mathcal{O}(1)}(\theta) = \widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha).$$

Proof. First assume that we have an $(\alpha, a, 0)$-block-encoding U of A. Using Lemma 2.11 we can implement a $(1, \mathcal{O}(a), \theta/256)$-block-encoding of $\sqrt{I+A/8}/4$ using the polynomial Q from Lemma 4.17 with $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha)$ uses of U. Applying this to ρ and measuring the last register yields 0 with probability $\frac{\theta}{256}$-close to

$$\mathrm{Tr}\left(\sqrt{I+A/8}/4\rho\sqrt{I+A/8}/4\right) = \mathrm{Tr}\big((I+A/8)\rho\big)/16 = 1/16 + \mathrm{Tr}\big(A\rho\big)/128.$$

By outputting 120 when there is a 0 in the last register and -8 otherwise we get a trace-estimator with bias less than $\theta/2$ and constant σ. To see this, call random variable corresponding to the output Z, and consider the case where we get a 0 exactly with probability $1/16 + \mathrm{Tr}\big(A\rho\big)/128$:

$$\begin{aligned}\mathbb{E}[Z] &= 120\cdot\big(1/16 + \mathrm{Tr}\big(A\rho\big)/128\big) - 8\cdot\big(1 - 1/16 - \mathrm{Tr}\big(A\rho\big)/128\big) \\ &= \frac{120}{16} - 8 + \frac{8}{16} + \frac{120}{128}\mathrm{Tr}\big(A\rho\big) + \frac{8}{128}\mathrm{Tr}\big(A\rho\big) \\ &= \mathrm{Tr}\big(A\rho\big).\end{aligned}$$

Note that an $\frac{\theta}{256}$ error in our probability can at most introduce a $\frac{120}{256}\theta \leq \frac{1}{2}\theta$ bias in our trace-estimator.

Now if we do have error in our block-encoding of A, i.e., if we have an $(\alpha, a, \theta/2)$-block-encoding of A, then we can view this as an $(\alpha, a, 0)$-block-encoding of a matrix $\tilde{A}$ that is $\frac{\theta}{2}$-close to A in operator norm. We know that $\left|\mathrm{Tr}\big(A\rho\big) - \mathrm{Tr}\big(\tilde{A}\rho\big)\right| \leq \left\|A - \tilde{A}\right\| \leq \theta/2$ and hence that a trace-estimator for $\tilde{A}$ with bias at most $\theta/2$ is a trace-estimator for A with bias at most θ. □

4.3.2 | Block-encodings for sparse matrices

In the last section we considered Gibbs-sampling and trace-estimating when A and H are given as block-encodings. If instead these matrices are given via a sparse matrix oracle, then we can use the following lemma to implement such block-encodings.

Lemma 4.19 *Suppose that A is an s-sparse operator in $[-1,1]^{n\times n}$, given in the sparse matrix input model (oracles* (4.6) *and* (4.7) *with respect to a fixed j), whose entries are all given as bitstrings of length B. Then we can implement an $(s, \widetilde{\mathcal{O}}_{\frac{n}{\varepsilon}}(B), \varepsilon)$-block-encoding of A with $\mathcal{O}(1)$ queries, and $\widetilde{\mathcal{O}}_{\frac{n}{\varepsilon}}(B)$ elementary operations.*[13]

Proof. The proof is implicit in the work of Childs [Chi10], a more explicit proof is given in [GSLW19, Lem. 48]. We fix an input matrix A and write $\mathbf{index}(k,i)$ for the index of the ith non-zero entry in the kth row of A. Let U_R be an algorithm that performs the mapping:[14]

$$\begin{aligned}
|k\rangle|0\rangle|0\rangle|0\rangle &\mapsto \frac{1}{\sqrt{s}}\sum_{i=0}^{s-1}|k\rangle|i\rangle|0\rangle|0\rangle \\
&\mapsto \frac{1}{\sqrt{s}}\sum_{i=0}^{s-1}|k\rangle|\mathbf{index}(k,i)\rangle|0\rangle|0\rangle \\
&= \frac{1}{\sqrt{s}}\sum_{a:A_{ka}\neq 0}|k\rangle|a\rangle|0\rangle|0\rangle \\
&\mapsto \frac{1}{\sqrt{s}}\sum_{a:A_{ka}\neq 0}|k\rangle|a\rangle|A_{ka}\rangle|0\rangle \\
&\mapsto \frac{1}{\sqrt{s}}\sum_{a:A_{ka}\neq 0}\sqrt{|A_{ka}|}|k\rangle|a\rangle|A_{ka}\rangle|0\rangle + \sqrt{1-|A_{ka}|}|k\rangle|a\rangle|A_{ka}\rangle|1\rangle \\
&\mapsto \frac{1}{\sqrt{s}}\sum_{a:A_{ka}\neq 0}\mathrm{sign}(A_{ka})\sqrt{|A_{ka}|}|k\rangle|a\rangle|A_{ka}\rangle|0\rangle + \sqrt{1-|A_{ka}|}|k\rangle|a\rangle|A_{ka}\rangle|1\rangle \\
&\mapsto \frac{1}{\sqrt{s}}\sum_{a:A_{ka}\neq 0}\mathrm{sign}(A_{ka})\sqrt{|A_{ka}|}|k\rangle|a\rangle|0\rangle|0\rangle + \sqrt{1-|A_{ka}|}|k\rangle|a\rangle|0\rangle|1\rangle.
\end{aligned}$$

Similarly, let U_C be an algorithm that performs the mapping

$$|\ell\rangle|0\rangle|0\rangle|0\rangle \mapsto \frac{1}{\sqrt{s}}\sum_{b:A_{b\ell}\neq 0}\sqrt{|A_{b\ell}|}|b\rangle|\ell\rangle|0\rangle|0\rangle + \sqrt{1-|A_{b\ell}|}|b\rangle|\ell\rangle|0\rangle|2\rangle.$$

[13] The dependence on s can be improved to roughly a square root when an upper bound on the operator norm of A is known [Low19].

[14] Here we ignore the fact that some rows might have less than s non-zero entries. If on a query ℓ the sparse oracle returns a symbol indicating that there is no ℓth non-zero element, then we can simply set the last register to one. Alternatively, we can assume that the sparse oracle always returns an index, but that the index may point to an entry that is zero.

We have

$$
\begin{aligned}
&\langle k|\langle 0|\langle 0|\langle 0|U_R^\dagger U_C|\ell\rangle|0\rangle|0\rangle|0\rangle\\
=&\left(\frac{1}{\sqrt{s}}\sum_{a:A_{ka}\neq 0}\operatorname{sign}(A_{ka})\sqrt{|A_{ka}|}\langle k|\langle a|\langle 0|\langle 0| + \sqrt{1-|A_{ka}|}\langle k|\langle a|\langle 0|\langle 1|\right)\\
&\cdot\left(\frac{1}{\sqrt{s}}\sum_{b:A_{b\ell}\neq 0}\sqrt{|A_{b\ell}|}|b\rangle|\ell\rangle|0\rangle|0\rangle + \sqrt{1-|A_{b\ell}|}|b\rangle|\ell\rangle|0\rangle|2\rangle\right)\\
=&A_{k\ell}/s.
\end{aligned}
$$

Hence, $U_R^\dagger U_C$ is a block-encoding of A that is subnormalized by a factor s. Note that both U_R and U_C can be implemented using $\mathcal{O}(1)$ queries. The indices used in the algorithm can be stored in $\mathcal{O}\big(\log(n)\big)$ qubits each, and a binary representation of $A_{k\ell}$ can be stored in B bits. The initial superposition over s elements can be set up with $\mathcal{O}\big(\log(n)\big)$ elementary operations. Using $\widetilde{\mathcal{O}}_{n/\varepsilon}(B)$ elementary operations we can add the amplitudes $\sqrt{|A_{ka}|}$ and $\sqrt{|A_{b\ell}|}$ with precision $\varepsilon/4n$ and add the sign of the entry as well. Now the (k,ℓ)-entry of our block-encoding is $2\frac{\varepsilon}{4n}\sqrt{A_{k\ell}}+\frac{\varepsilon^2}{16n^2}\leq\varepsilon/n$ precise. This implies that the block-encoding is an ε-approximate block-encoding of A. □

Although not needed for our applications, the above proof can easily be extended to asymmetric complex matrices if we assume access to sparse oracles for both the rows and columns, and by replacing $\operatorname{sign}(A_{ka})$ with a phase corresponding to the complex argument of A_{ka}.

4.4 | An SDP-solver in the sparse matrix input model

To obtain a quantum SDP-solver for the sparse matrix input model it now remains to show how to create a sparse matrix oracle for $H^{(t)}$. After we do this we will put out different subroutines together to implement an SDP-solver in the sparse matrix input model. We conclude this section with a lower bound on sparse matrix summation, showing that other techniques will be needed in order to speedup SDP-solving in the sparse matrix input model further.

4.4.1 | Implementing a sparse oracle for H

To keep this section general we simplify the notation: let H be the sum of d different s-sparse matrices $M_i \in \mathbb{R}^{n\times n}$ as follows

$$H=\sum_{i=1}^{d} M_i.$$

In this section we study the complexity of implementing one sparse matrix oracle call to H (denoted by O_H^{sparse}), given access to respective sparse matrix oracles for the matrices

$M_1, \ldots, M_d$. We will only focus on the oracle that computes the non-zero indices of H, since the oracle that gives element-access is easy to compute by summing the results of queries to the separate oracles, which uses $\mathcal{O}(d)$ queries.

In the remainder of this section we only consider one row of H. We denote the list of indices of non-zero entries in this row by R_H, and the corresponding lists for the rows of the matrices M_i by R_i. Notice that each of the R_i is an ordered list of at most s integers, and that R_H will again be an ordered list of integers, containing all integers in the R_i lists once (i.e., R_H does not contain duplicates).

We first show that an oracle for R_H can be constructed efficiently classically. The important observation is that in the classical case we can write down the oracle once and store it in memory. Hence, we can create an oracle for R_H as follows. We start from the oracle of R_1 and then we "add" the oracle of R_2, then that of R_3, and so on. By this we mean that we start with an empty list and insert the integers from R_1 into the list, then we insert the integers from R_2 that are not present yet into this list, and so on. When an efficient data structure (for example a binary heap) is used, then each of such single integer insertions can be done using polylog(d, s, n) time. This shows that in the classical case we can construct an oracle for R_H in time $\widetilde{\mathcal{O}}_n(sd)$. Multiplying this complexity with the number of rows gives us an oracle for H in time $\widetilde{\mathcal{O}}(nsd)$. Note that we only need to compute this oracle once, after which queries can be performed in time $\widetilde{\mathcal{O}}_{sdn}(1)$. In the application we are interested in, Meta-Algorithm 4.1, in each iteration t only one new matrix $M^{(t)}$ 'arrives', hence from the oracle for $H^{(t-1)}$ an oracle for $H^{(t)}$ can be constructed in time $\widetilde{\mathcal{O}}(ns)$.

The quantum case is similar, but we need to add all the matrices together each time a query to O_H^{sparse} is made, since writing down each row of H in every iteration would take $\Omega(n)$ operations, which we cannot afford. Apart from this change we can use the same technique as for the classical case to compute a list of the non-zero entries in a row of H. This list allows us to compute the **index** function once, after which we uncompute the list again. Given the index of the ℓth non-zero entry in a row, we can use the same approach to calculate ℓ. It follows that we can perform these calculations "in place" as well.

4.4.2 | Putting everything together

Now that we know how to construct a sparse oracle for H, we can give an upper bound on the complexity of Gibbs-sampling in the sparse matrix input model.

Lemma 4.20 *In the sparse matrix input model for s-sparse $n \times n$ matrices*

$$T_{Gibbs}(K, d, \theta) = \widetilde{\mathcal{O}}_{\frac{K}{\theta}}\left(\sqrt{n} d^2 s^2 K\right).$$

Proof. Using the argument from Section 4.4.1 we can implement an oracle for the non-zero entries of the sum of d matrices that are s-sparse using $\mathcal{O}(ds)$ queries. An oracle for the values of the entries can be implemented with $\mathcal{O}(d)$ queries and $\widetilde{\mathcal{O}}_{\frac{K}{\theta}}(d)$ elementary operations by summing the values from the separate oracles.

Since H is ds-sparse we can use Lemma 4.19 to implement a block-encoding for H. However, the entries of H can be as large as K, so instead we implement a block-encoding for H/K by scaling down the results from a query to the value oracle for H. The result will still be a block-encoding of H, but with a larger scale factor. We end up with a $\left(dsK, \widetilde{\mathcal{O}}_{\frac{Kdsn}{\theta}}(1), \widetilde{\mathcal{O}}\left(\frac{\theta^2}{Kdsn^2}\right)\right)$-block-encoding of H using $\mathcal{O}(1)$ queries to the oracle for H (and hence $\mathcal{O}(ds)$ queries to the input) and $\widetilde{\mathcal{O}}_{\frac{ndK}{\theta}}(1)$ elementary operations. The lemma then follows from Lemma 4.15. □

This leads to the following upper bound on quantum SDP-solving.

Theorem 4.21 *In the sparse matrix input model*

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\gamma^8\right),$$

where $\gamma = Rr/\varepsilon$. For an SDP primal oracle the same complexity can be accomplished with the choice $\gamma = R/\varepsilon$.

Proof. Using Lemma 4.20 for $K = \gamma$, $d = \gamma^2$ and $\theta = \gamma^{-1}$ shows that in the sparse matrix input model

$$T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1}) = \widetilde{\mathcal{O}}_{\frac{K}{\theta}}\left(\sqrt{n}s^2\gamma^5\right).$$

Using Lemma 4.18 with a block-encoding from Lemma 4.19 we get

$$T_{Tr}^{\mathcal{O}(1)}(\gamma^{-1}) = \widetilde{\mathcal{O}}_{\gamma}(s).$$

It follows from Theorem 4.13 that

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\gamma^8\right).$$ □

4.4.3 | A lower bound on sparse matrix summation

A large contribution to the complexity of the SDP-solver from the last section is the cost of implementing a sparse oracle for H. In this section we prove that such a sparse oracle cannot be implemented more efficiently.

We show a lower bound on the query complexity of implementing the oracle O_H^{sparse} by observing that determining the number of elements in a row of H solves the majority function. Notice that, given access to O_H^{sparse}, we can decide with a single query whether there are at least a certain number of non-zero elements in a row of H.

Lemma 4.22 *Given $d+1$ ordered lists of integers $R_0, \ldots, R_d$, each of length at most s. Let R_H be the merged list that is ordered and contains every element in the lists R_i only once (i.e., we remove duplicates). Deciding whether $|R_H| \leq s + \frac{sd}{2}$ or $|R_H| \geq s + \frac{sd}{2} + 1$ takes $\Omega(sd)$ quantum queries to the input lists in the worst case.*

Proof. We prove this by a reduction from MAJ on sd elements. Let $Z \in \{0,1\}^{s\times d}$ be a Boolean string. It is known that it takes at least $\Omega(sd)$ quantum queries to Z to decide whether $|Z| \leq \frac{sd}{2}$ or $|Z| \geq \frac{sd}{2}+1$ [BBCMW01]. Now let $R_0, R_1, \dots, R_d$ be lists of length s defined as follows:

- $R_0[j] = j(d+1)$ for $j = 1, \dots, s$.
- $R_i[j] = j(d+1) + jZ_{ij}$ for $j = 1, \dots, r$ and $i = 1, \dots, d$.

By construction, if $Z_{ij} = 1$, then the value of the entry $R_i[j]$ is unique in the lists $R_0, \dots, R_d$, and if $Z_{ij} = 0$, then $R_i[j] = R_0[j]$. So in R_H there will be one element for each element in R_0 and one element for each bit in Z_{ij} that is 1. The length of R_H is therefore $s + |Z|$. Hence, distinguishing between $|R_H| \leq s + \frac{sd}{2}$ and $|R_H| \geq s + \frac{sd}{2} + 1$ would solve the MAJ problem and therefore requires at least $\Omega(sd)$ queries to the lists in the worst case. □

Corollary 4.23 *Implementing a query to a sparse matrix oracle O_H^{sparse} for*

$$H = \sum_{j=i}^{d} M_j,$$

where each M_j is s-sparse, requires $\Omega(sd)$ queries to the $O_{M_j}^{sparse}$ in the worst case.

In the next section we show how to avoid implementing the sparse matrix queries for H in order to speed up the quantum SDP-solver further.

4.5 | An SDP-solver in the quantum operator input model

To implement an SDP-solver in the quantum operator input model it remains to show how to implement a block-encoding of $H = \sum_{j=0}^{m} y_j A_j$ efficiently. This is done by using an efficient data structure as the intermediate storage of y. This data structure has recently been used by many quantum algorithms as part of their input model. However, as far as we are aware this is the first use of it as an intermediate storage in a quantum algorithm. Combining these techniques for block-encodings with the meta-algorithm of Theorem 4.33 leads to an efficient quantum SDP-solver for the quantum operator input model, see Theorem 4.34. This SDP-solver can also be used to implement an improved SDP-solver in the sparse matrix input model. Before we implement an SDP-solver in the quantum operator input model we first motivate this model further by showing that many other input models can be used to implement block-encodings of the input matrices.

4.5.1 | Other input models

There are multiple input models that reduce to the quantum operator input model. In this section we give three examples.

Quantum state input model The *quantum state input model* is inherently quantum and has no classical counterpart for SDPs. A version of this model was first introduced by Brandão and Svore [BS17]. In this model we assume that each A_j has a fixed decomposition of the form

$$A_j = \mu_j^+ \varrho_j^+ - \mu_j^- \varrho_j^- + \mu_j^I I$$

for (subnormalized) density operators $\varrho_j^\pm$, non-negative reals $\mu_j^\pm$ and real number $\mu_j^I \in \mathbb{R}$. We assume access to an oracle O_μ that takes as input an index j and outputs binary representations[5] of μ_j^+, μ_j^- and μ_j^I.

Furthermore we assume access to a state-preparing oracle $O_{|\cdot\rangle}$ that prepares purifications (see Definition 2.2)[15] $\left|\psi_j^\pm\right\rangle$ of $\varrho_j^\pm$:

$$O_{|\cdot\rangle}|j\rangle|\pm\rangle|0\rangle = \left|j\right\rangle\left|\pm\right\rangle\left|\psi_j^\pm\right\rangle.$$

Here $|+\rangle$ or $|-\rangle$ are some binary descriptions of "+" or "-", not the states $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$.[16] Finally we assume that a bound $B \in \mathbb{R}_{>0}$ is known such that

$$\forall j : \mu_j^+ + \mu_j^- \leq B.$$

Note that a tight upper bound B can easily be found using $\mathcal{O}\left(\sqrt{m}\right)$ quantum queries to O_μ by means of maximum-finding [DH96].

Lemma 4.24 : [LC16] *Let G be a $(w+a)$-qubit unitary which on the input state $|0\rangle^{\otimes w}|0\rangle^{\otimes a}$ prepares a purification $\left|\varrho\right\rangle$ of the subnormalized w-qubit density operator ϱ. Then we can implement a $(1, \mathcal{O}(w+a), 0)$-block-encoding of ϱ, with single use of G and $G^\dagger$, and with $\mathcal{O}(w+a)$ elementary operations.*

Proof. Let

$$\varrho = \sum_{i=1}^{2^w} \lambda_i |\psi_i\rangle\langle\psi_i|$$

be a spectral decomposition of ϱ such that the purification $\left|\varrho\right\rangle$ is

$$\left|\varrho\right\rangle = \sum_{i=1}^{2^w} \sqrt{\lambda_i}|\psi_i\rangle|0\rangle|\phi_i\rangle + |\perp\rangle,$$

where $|\perp\rangle$ has no overlap with the $|0\rangle$ state in the second register and where $\left\langle\phi_i\middle|\phi_j\right\rangle = \delta_{ij}$. Without loss of generality we assume that this second register is a single qubit, since if it would be multiple qubits then we could compute the OR of these bits in a new register and use that single qubit instead. Hence $|\perp\rangle = \sum_{i=1}^{2^w} \alpha_i \left|\tilde{\psi}_i\right\rangle\left|1\right\rangle\left|\tilde{\phi}_i\right\rangle$.

[15] For simplicity we assume that for a d-dimensional density operator a purification has at most polylog(d) qubits.

[16] Technically any pair of easy-to-prepare orthogonal states would work, and hence the states $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ could be used as well.

Consider the state

$$\begin{aligned}
&\big(I_{2^{w+1}} \otimes G\big)\big|\psi_i\big\rangle\big|0^{w+a+1}\big\rangle\\
=&\big|\psi_i\big\rangle\big|0\big\rangle\left(\sum_{k=1}^{2^w}\sqrt{\lambda_k}\big|\psi_k\big\rangle\big|0\big\rangle\big|\phi_j\big\rangle + |\perp\rangle\right)\\
=&\sum_{k=1}^{2^w}\sqrt{\lambda_k}\big|\psi_i\big\rangle\big|0\big\rangle\big|\psi_k\big\rangle\big|0\big\rangle\big|\phi_k\big\rangle + \sum_{k=1}^{2^w}\alpha_k\big|\psi_i\big\rangle\big|0\big\rangle\big|\tilde{\psi}_k\big\rangle\big|1\big\rangle\big|\tilde{\phi}_k\big\rangle,
\end{aligned}$$

and the state (where SWAP_{w+1} swaps the first $w+1$ qubits with the second $w+1$ qubits)

$$\begin{aligned}
&(\mathrm{SWAP}_{w+1} \otimes I_{2^a})\big(I_{2^{w+1}} \otimes G\big)\big|\psi_j\big\rangle\big|0^{w+a+1}\big\rangle\\
=&\sum_{\ell=1}^{2^w}\sqrt{\lambda_\ell}\big|\psi_\ell\big\rangle\big|0\big\rangle\big|\psi_j\big\rangle\big|0\big\rangle\big|\phi_\ell\big\rangle + \sum_{\ell=1}^{2^w}\alpha_\ell\big|\tilde{\psi}_\ell\big\rangle\big|1\big\rangle\big|\psi_j\big\rangle\big|0\big\rangle\big|\tilde{\phi}_\ell\big\rangle.
\end{aligned}$$

The inner product between these two states is

$$\begin{aligned}
&\big\langle\psi_i\big|\big\langle 0^{w+a+1}\big|\Big(I_{2^{w+1}} \otimes G^\dagger\Big)(\mathrm{SWAP}_{w+1} \otimes I_{2^a})\big(I_{2^{w+1}} \otimes G\big)\big|\psi_j\big\rangle\big|0^{w+a+1}\big\rangle\\
=&\left(\sum_{k=1}^{2^w}\sqrt{\lambda_k}\big\langle\psi_i\big|\big\langle 0\big|\big\langle\psi_k\big|\big\langle 0\big|\big\langle\phi_k\big|\right)\left(\sum_{\ell=1}^{2^w}\sqrt{\lambda_\ell}\big|\psi_\ell\big\rangle\big|0\big\rangle\big|\psi_j\big\rangle\big|0\big\rangle\big|\phi_\ell\big\rangle\right)\\
=&\sum_{k=1}^{2^w}\sum_{\ell=1}^{2^w}\sqrt{\lambda_k\lambda_\ell}\big\langle\psi_i\big|\psi_\ell\big\rangle\big\langle\psi_k\big|\psi_j\big\rangle\big\langle\phi_k\big|\phi_\ell\big\rangle\\
=&\sum_{k=1}^{2^w}\lambda_k\big\langle\psi_i\big|\psi_k\big\rangle\big\langle\psi_k\big|\psi_j\big\rangle\\
=&\sum_{k=1}^{2^w}\lambda_k\delta_{ik}\delta_{jk}\\
=&\lambda_i\delta_{ij}
\end{aligned}$$

Hence $V := \big(I_{2^{w+1}} \otimes G^\dagger\big)(\mathrm{SWAP}_{w+1} \otimes I_{2^a})\big(I_{2^{w+1}} \otimes G\big)$ is a $(1,\mathcal{O}(w+a),0)$-block-encoding of $\mathrm{diag}(\lambda)$ in the $\{|\psi_i\rangle\}$ basis, which is exactly a block-encoding of ϱ. □

This lemma shows that we can reduce the quantum state input model to the quantum operator input model with $\alpha = B$. Brandão et al. [BKLLSW19] showed that in the quantum state input model the dependence on n for Gibbs-sampling that we would get via such a reduction can be replaced with a dependence on the maximal rank of the input matrices, which might be better in certain settings. We later showed in [AG19a] that the dependence on the rank can be replaced by a dependency on B. However, in this chapter we will only consider SDP-solvers for the quantum state input model that are derived from SDP-solvers for the quantum operator input model, and hence we will only consider SDP-solvers for this model with a $\sqrt{n}$ dependence on the dimension.

Hamiltonian input model In the *Hamiltonian* input model for SDPs, we have access to two oracles for the A_j matrices. The first oracle, O_t, gives a classical description of a real vector $t \in \mathbb{R}^{m+1}$ in the usual way

$$O_t|j\rangle|0\rangle = |j\rangle|t_j\rangle.$$

The second oracle, O_{Hsim}, performs Hamiltonian simulation with A_j and time parameter $1/t_j$:

$$O_{Hsim}|j\rangle|\psi\rangle = |j\rangle e^{\mathbf{i}A_j/t_j}|\psi\rangle.$$

Alongside the oracles we also require an upper bound

$$\tau \geq \max_j t_j$$

as part of the input for an SDP. As in the other input models, we assume that we can also apply the inverse of the oracles. Consider the following lemma:

Lemma 4.25 : [GSLW19, Cor. 71] *Suppose that $U = e^{\mathbf{i}A/t}$, where A is a Hamiltonian of norm at most $t/2$. Let $\varepsilon \in (0,1/2]$, then we can implement a $(2t/\pi,2,\varepsilon)$-block-encoding of A with $\mathcal{O}\big(\log(t/\varepsilon)\big)$ uses of controlled-U and its inverse, using $\mathcal{O}\big(\log(t/\varepsilon)\big)$ elementary operations and using a single ancilla qubit.*

If all t_j are at least 2, then this lemma shows that we can implement the quantum operator input model using the Hamiltonian input model with α = 2τ. Assuming such a lower bound on the t_j is not a significant limitation. In order to be able to distinguish $e^{\mathbf{i}A_j/t_j}$ from $e^{-\mathbf{i}A_j/t_j}$ we need that the eigenvalues of A_j/t_j are not too close to π, and hence we want a constant lower bound on t_j anyway.

POVM input model Similarly to Low and Chuang [LC17], we note that if one can perform a POVM measurement on a quantum computer, then one can also implement a block-encoding of the corresponding measurement operator. First we clarify what we mean by performing a POVM measurement on a quantum computer. For simplicity assume that the POVM is a two-outcome measurement, represented by the operators $M,(I-M)$ that correspond to outcomes 0 and 1 respectively. Then we assume (without too much loss of generality) that an implementation on a quantum computer is as follows: we get as input a mixed state ρ, and attach a ancilla qubits to it. Then we apply some unitary on the state, and finally perform a measurement of the last qubit in the computational basis, accepting only measurement outcomes where the last qubit is $|0\rangle$.

In the POVM input model we assume that $0 \preceq A_j$ and that we have access to a unitary that performs the measurement under A_j as above. Note that this is very similar (although not exactly equal) to assuming access to an unbiased trace-estimator. Using such an input model we can also implement block-encodings of the input matrices.

Lemma 4.26 *Let $\{M, I-M\}$ be a POVM measurement. Suppose we have access to a unitary U such that*

$$\forall\rho : \left|\mathrm{Tr}\Big((I\otimes|0\rangle\langle 0|)U\big(\rho\otimes|0\rangle\langle 0|^{\otimes a}\big)U^\dagger\Big)-\mathrm{Tr}\big(M\rho\big)\right|\leq\varepsilon. \tag{4.9}$$

Then we can implement a $(1, a+1, \varepsilon)$-block-encoding of M using a single query to U and $U^\dagger$ and a single CNOT gate.

Proof. Observe that $\rho\otimes|0\rangle\langle 0|^{\otimes a}=\big(I\otimes|0\rangle^{\otimes a}\big)\rho\big(I\otimes\langle 0|^{\otimes a}\big)$, and thus by the cyclic nature of the trace

$$\mathrm{Tr}\Big((I\otimes|0\rangle\langle 0|)U\big(\rho\otimes|0\rangle\langle 0|^{\otimes a}\big)U^\dagger\Big)=\mathrm{Tr}\Big(\big(I\otimes\langle 0|^{\otimes a}\big)U^\dagger(I\otimes|0\rangle\langle 0|)U\big(I\otimes|0\rangle^{\otimes a}\big)\rho\Big).$$

Therefore we can rewrite (4.9) as

$$\begin{aligned}
\forall\rho: \qquad & \left|\mathrm{Tr}\Big((I\otimes|0\rangle\langle 0|)U\big(\rho\otimes|0\rangle\langle 0|^{\otimes a}\big)U^\dagger\Big)-\mathrm{Tr}\big(M\rho\big)\right|\leq\varepsilon\\
\Longleftrightarrow\forall\rho: \qquad & \left|\mathrm{Tr}\Big(\Big[\big(I\otimes\langle 0|^{\otimes a}\big)U^\dagger(I\otimes|0\rangle\langle 0|)U\big(I\otimes|0\rangle^{\otimes a}\big)-M\Big]\rho\Big)\right|\leq\varepsilon\\
\Longleftrightarrow \qquad & \left\|\big(I\otimes\langle 0|^{\otimes a}\big)U^\dagger(I\otimes|0\rangle\langle 0|)U\big(I\otimes|0\rangle^{\otimes a}\big)-M\right\|\leq\varepsilon.
\end{aligned}$$

Finally we can postpone a measurement by copying the result to an extra qubit, so

$$\begin{aligned}
&\big(I\otimes\langle 0|^{\otimes a}\big)U^\dagger(I\otimes|0\rangle\langle 0|)U\big(I\otimes|0\rangle^{\otimes a}\big)\\
&=\big(I\otimes\langle 0|^{\otimes a+1}\big)(U^\dagger\otimes I)(I\otimes\mathrm{CNOT})(U\otimes I)\big(I\otimes|0\rangle^{\otimes a+1}\big).
\end{aligned}$$

We conclude that $\big(U^\dagger\otimes I\big)(I\otimes\mathrm{CNOT})(U\otimes I)$ is a $(1, a+1, \varepsilon)$-block-encoding of M. □

4.5.2 | Implementing an SDP-solver

In this subsection we show how to efficiently implement a block-encoding for a linear combination of the input matrices by storing $y^{(t)}$ in an efficient data structure. Doing so by-passes the entrywise summation of the input matrices which was a major bottleneck in the sparse matrix input model. Now we describe how to use a quantum-access classical RAM (QCRAM) to efficiently implement a state-preparation-pair unitary that can be used to construct a linear combination of the block-encodings.

Lemma 4.27 *There is a data structure that can store an m-dimensional d-sparse vector $y\in\mathbb{R}^m$ with θ-precision in ℓ_1-norm using a QCRAM of size $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(d)$ such that:*

- *Initializing the data structure can be done with $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1)$ classical operations.*
- *Given a classical s-sparse vector, adding[17] it to the stored vector has classical cost $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(s)$.*

[17] In order to avoid error accumulation from repeated rounding, we assume for simplicity that there can be at most $\mathcal{O}\big(\mathrm{poly}(m/\theta)\big)$ such calls to the data structure in total.

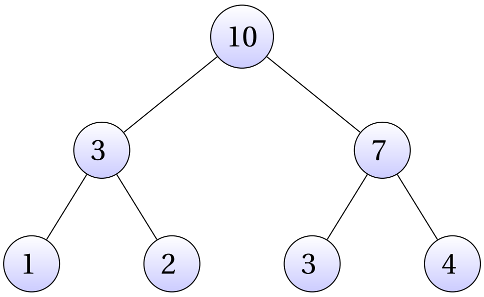


Figure 4.3: Tree structure for the vector $(1,2,3,4)$

- *Querying the ith element of the stored vector can be done using $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1)$ elementary operations.*
- *Given a $\beta \geq \|y\|_1$ we can implement a (symmetric) $\left(\beta, \widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1), \theta\right)$-state-preparation pair for y with $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1)$ elementary operations.*

Proof. We use the data structure of [KP17, App. A], we will give a rough sketch here, see their paper for more details. This data structure stores a vector on the leaves of a binary tree. The sum of the branches is stored at each vertex, see Figure 4.3 for an example. Now, updating one entry can be done by walking the tree upward and changing each node as required, which takes $\mathcal{O}(\ln(m))$ classical operations. Hence, an s-sparse vector can be added using $\mathcal{O}(s\ln(m))$ classical operations. Given a tree structure for a vector y, sampling from $y/\|y\|_1$ can easily be implemented classically: walk down the tree, going left or right with probabilities proportional to the node values. With a similar method a state-preparation pair for y can be implemented. Finally we note that there is no need to initialize a full-sized empty tree, the tree can be extended every time new entries are added. □

Corollary 4.28 *Having access to the above data structure for y, we have $T_{\text{Gibbs}}(K,d,\theta) = \widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha K\sqrt{n})$ in the quantum operator input model with parameter α.*

Proof. This follows from applying the Gibbs-sampling of Lemma 4.15 to the block-encoding of $H = \sum_{j=0}^{m} y_j A_j$ that is obtained by using Lemma 2.9 with the state-preparation pair from Lemma 4.27. □

This directly gives the following result for SDP-solving:

Theorem 4.29 *In the quantum operator input model with parameter α*

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}\left(\sqrt{mn}\alpha\gamma^4\right),$$

where $\gamma = Rr/\varepsilon$. For an SDP primal oracle the same complexity can be accomplished with $\gamma = R/\varepsilon$. The input oracle of the quantum operator input model can be implemented using $\mathcal{O}(1)$ queries and $\widetilde{\mathcal{O}}_{mn\alpha\gamma}(1)$ elementary operations in the sparse matrix input model with setting $\alpha = s$.[18]

Proof. The complexity statement follows from Theorem 4.13 using Lemma 4.18 and Corollary 4.28. The reduction of the sparse matrix input model to the quantum operator input model follows from Lemma 4.19. □

4.6 | Two-Phase Quantum Search and Minimum-Finding

To further speed up the SDP-solvers derived from the meta-algorithms in Section 4.2 we can use a fast version of the Quantum OR Lemma, as in [BKLLSW19]. They prove the following lemma, see also [GSLW19]:

Lemma 4.30 : Fast Quantum OR Lemma [BKLLSW19] *Let $\Pi_1,\dots,\Pi_m$ be projectors and ρ a quantum state. Suppose that either*

1. *$\exists j$ s.t. $\mathrm{Tr}(\Pi_j\rho) \geq 1-\delta_1$, or*
2. *$\frac{1}{m}\sum_{j=1}^m \mathrm{Tr}(\Pi_j\rho) \leq \delta_2$*

for some $0 < \delta_1 \leq 1/2$ and $0 < \delta_2 \leq \frac{(1-\delta_1)^2}{12m}$. Then for all $\xi \in (0,1)$ there is a procedure that accepts with probability at least $(1-\delta_1)^2/4-\xi$ in the first case, and probability at most $3m\delta_2+\xi$ in the second case, and that uses 1 copy of ρ and $\mathcal{O}\left(\sqrt{m}/\xi\right)$ applications of a controlled version of the reflection $I-2\sum_{j=1}^m \Pi_j\otimes|j\rangle\langle j|$, and $\mathcal{O}\left(\sqrt{m}\frac{\mathrm{polylog}(m)}{\xi}\right)$ elementary operations.

This lemma is almost the same as the original Quantum OR Lemma [HLM17] but with the addition that the algorithm requires only $\mathcal{O}\left(\sqrt{m}/\xi\right)$ applications of the controlled reflection, compared to a linear number of reflections required by the original. In a recent paper Aaronson [Aar18] proved the *Gentle Quantum Search Lemma* using the Quantum OR Lemma. His proof can easily be extended to use the Fast Quantum OR Lemma. We call the more efficient version that we get as a result *Two-Phase Quantum Search.*

In the setting of the Two-Phase Quantum Search we have m procedures for decision problems and we ask whether one of them evaluates to 1, and if so, we want to find one that does. We also know that all procedures start with preparing some state ρ, followed by some procedure U_j that depends on the index of the decision problem $j \in [m]$. In classical deterministic processes it is quite natural that only one preparation of ρ is needed since the result can be copied and reused. For bounded-error classical processes $\mathcal{O}(\log(m))$ preparations of ρ suffice to get the error probability of one decision

[18] As mentioned in footnote 13 on page 86, this can be improved to $\alpha = \sqrt{s}$.

problem below $\frac{1}{3m}$. By the classical union bound this is low enough that we can find a marked element with constant probability. However, if ρ is a quantum state and the U_j are quantum algorithms, then such a bound is not so straightforward, since progress made in constructing ρ might be destroyed when running one of the U_j. Nevertheless, using the Fast Quantum OR Lemma it can be shown that $\widetilde{\mathcal{O}}\left(\log(m)^4\right)$ samples from ρ suffice to decide if one of the m procedures evaluate to 1.

Lemma 4.31 : Two-Phase Quantum Search *Let $\mu \in (0,1)$. Let ρ be a quantum state and $U_1,\ldots,U_m$ be unitaries with the U_j accessible through a unitary U that acts as $U|j\rangle|\psi\rangle = |j\rangle U_j|\psi\rangle$. Then there is a quantum algorithm that using $\widetilde{\mathcal{O}}\left(\log(m)^4\log(1/\mu)\right)$ samples of ρ and $\widetilde{\mathcal{O}}\left(\sqrt{m}\log(1/\mu)\right)$ applications of U and $U^\dagger$, outputs with success probability at least $1-\mu$ either*

- *a j such that $\mathrm{Tr}\left((I\otimes|1\rangle\langle 1|)U_j\rho U_j^\dagger\right) \geq 1/3$, i.e., a j such that U_j outputs 1 with probability at least $1/3$ on input ρ,*
- *or concludes correctly that $\mathrm{Tr}\left((I\otimes|1\rangle\langle 1|)U_j\rho U_j^\dagger\right) < 2/3$ for all j, i.e., no unitary outputs 1 with probability at least $2/3$ on input ρ.*

Proof. This follows directly from the proof of the Gentle Quantum Search Lemma in [Aar18, Lem. 15], by replacing the normal Quantum OR Lemma by the Fast Quantum OR Lemma [BKLLSW19]. □

Using the above lemma we construct the *Two-Phase Quantum Minimum-Finding* algorithm. It turns out that we need to use this algorithm in a situation where different values have different error-bars, therefore the statement gets slightly complicated. In other use-cases one can typically just choose each error-margin η_i equal to δ resulting in a simpler statement.

Lemma 4.32 : Two-Phase Quantum Minimum-Finding *Let $\delta,\mu' \in (0,1)$. Let ρ be a quantum state and $U_1,\ldots,U_m$ be unitaries, with the U_j accessible through a unitary U that acts as $U|j\rangle|\psi\rangle = |j\rangle U_j|\psi\rangle$. Let $a_1,\ldots,a_m$, $\eta_1,\ldots,\eta_m$ be real numbers such that $\min_j|a_j|+|\eta_j| \leq M$. Assume that with probability at least $2/3$, U_j computes a binary representation of a_j up to additive error η_j using one copy of ρ. Then, with probability at least $1-\mu'$, we can find a j such that $a_j-\eta_j \leq \min_i(a_i+\eta_i)+\delta$ using $\widetilde{\mathcal{O}}\left(\log(m)^4\log(M/\delta)\log(1/\mu')\right)$ samples of ρ and $\widetilde{\mathcal{O}}\left(\sqrt{m}\log(1/\mu')\log(M/\delta)\right)$ applications of U and its inverse.*

Proof. Do a binary search on the value v to precision δ by checking whether there is still an element with $a_i+\eta_i \leq v$ using Lemma 4.31 in each round with setting $\mu = \Theta\left(\mu'/\log(M/\delta)\right)$. This binary search will result in a value $v \leq \min_i(a_i+\eta_i)+\delta$ with probability at least $1-\mu'/2$, and it is not hard to see that the last j found by the algorithm from Lemma 4.31 during the binary search will be such that $a_j-\eta_j \leq v$ with probability at least $1-\mu/2 \geq 1-\mu'/2$. Therefore this j satisfies the required inequality with probability at least $1-\mu'$. □

This leads to the following improved bound on SDP-solving.

Theorem 4.33 *Assume that $y^{(t)} \in \mathbb{R}^{m+1}$ is stored in a data structure such that updating an entry requires at most $\widetilde{\mathcal{O}}\big(T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\big)$ elementary operations, where $\gamma := 6Rr/\varepsilon$. Furthermore, assume that both $T_{Gibbs}\big(\gamma, \gamma^2, \gamma^{-1}\big)$ and $T_{Tr}^{\sigma}\big(\gamma^{-1}\big)$ are at most polynomial in their parameters. Then*

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}_n\Big(\big(\sqrt{m}T_{Tr}^{\sigma}(\gamma^{-1}) + T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\big)\gamma^4\sigma^2\Big).$$

Similarly there is also a quantum algorithm with the same complexity, but with $\gamma := 6R/\varepsilon$, that implements an SDP primal oracle.

Proof. To obtain an ε-approximate SDP-solver we use binary search and the Arora-Kale framework to find an $\varepsilon/4$-approximation of OPT. After we found such an approximation $\widetilde{\mathrm{OPT}}$, we can run the Arora-Kale algorithm with guess $\omega = \widetilde{\mathrm{OPT}} + \frac{3}{4}\varepsilon$ and precision $\varepsilon/4$ to find a feasible dual solution that is ε-optimal. Finally, we run the SDP primal oracle with $\omega = \widetilde{\mathrm{OPT}} - \frac{3}{4}\varepsilon$ and precision $\varepsilon/4$ in order to obtain a primal solution that is ε-feasible and has an objective value that is ε-close to optimal.

The two meta-algorithms each run for $\widetilde{\mathcal{O}}_n\big(\gamma^2\big)$ iterations. In each iteration we need to update at most three entries in the data structure for the y vector, which takes at most $\widetilde{\mathcal{O}}\big(T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\big)$ elementary operations by assumption. To search for a violated constraint when using the SDP primal oracle framework, we use the Two-Phase Quantum Search, and we use Two-Phase Quantum Minimum-Finding to implement the minimum-finding needed in the advisor for the Arora-Kale framework[19], following the geometric approach of Lemma 4.8 for implementing a γ^{-1}-advisor.

Consider $\widetilde{\rho}^{\otimes k}$, where $\widetilde{\rho}$ is a density operator that is a $\frac{1}{4\gamma}$-approximation in trace distance of the Gibbs state ρ corresponding to the current y vector and where $k = 3(4\sigma\gamma)^2$. Let U_j be the operator that applies a $(\frac{1}{4\gamma}, \sigma)$-trace-estimator to each copy of $\widetilde{\rho}$ and takes the average of the outcomes, i.e., it obtains estimates of $\mathrm{Tr}\big(A_j\widetilde{\rho}\big)$ with bias at most $\frac{1}{4\gamma}$ and standard deviation at most σ independently k times, taking the average at the end. By Chebyshev's inequality we can see that this way U_j computes a $\frac{1}{2\gamma}$-precise estimate of $\mathrm{Tr}\big(A_j\widetilde{\rho}\big)$ with probability at least $2/3$. We have that $\big\|\widetilde{\rho} - \rho\big\|_{tr} \leq \frac{1}{4\gamma}$ and thus we get a $\frac{3}{4\gamma}$-precise estimate of $\mathrm{Tr}\big(A_j\rho\big)$ with probability at least $\frac{2}{3}$. The preparation of $\widetilde{\rho}^{\otimes k}$ can be performed using $kT_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})$ queries by definition, while U_j can be implemented with query complexity $kT_{Tr}^{\sigma}(\gamma^{-1})$.

By using Two-Phase Quantum Search (Lemma 4.31) and Two-Phase Quantum Minimum-Finding (Lemma 4.32), with $\mu = \widetilde{\mathcal{O}}_n\big(\gamma^{-2}\big)$ and with $\widetilde{\rho}^{\otimes k}$ as the initially prepared quantum state (called ρ in those lemmas), we can implement each iteration of the meta-algorithms using $\widetilde{\mathcal{O}}_n\Big(\big(\sqrt{m}T_{Tr}^{\sigma}(\gamma^{-1}) + T_{Gibbs}(\gamma, \gamma^2, \gamma^{-1})\big)\gamma^2\sigma^2\Big)$ queries. The

[19] In the advisor implementation of Lemma 4.8 the minimum-finding is not done over the computed traces, but rather the angles calculated using these traces. Precision $\delta = 1/\mathrm{poly}(\gamma)$ suffices for the trace → angle conversion, and since the magnitude M of angles is bounded by π, we get $\log(M/\delta) = \mathcal{O}\big(\log(\gamma)\big)$ in Lemma 4.32.

stated final complexity follows by multiplying the complexity per iteration with the number of iterations $T = \widetilde{\mathcal{O}}_n(\gamma^2)$. □

The following theorem then follows directly.

Theorem 4.34 *In the quantum operator input model*

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}\big((\sqrt{m} + \sqrt{n}\gamma)\alpha\gamma^4\big),$$

where $\gamma = Rr/\varepsilon$. For an SDP primal oracle the same complexity can be accomplished with $\gamma = R/\varepsilon$. The input oracle of the quantum operator input model can be implemented using $\mathcal{O}(1)$ queries and $\widetilde{\mathcal{O}}_{mn\alpha\gamma}(1)$ elementary operations in the sparse matrix input model with $\alpha = s$.[18]

A natural question to ask is whether the dependence on all the parameters is optimal. In Section 9.2.5 we prove an $\widetilde{\Omega}\big((\sqrt{m} + \sqrt{n})\alpha\gamma\big)$ lower bound. Hence, apart from the dependence on γ, our upper bound is tight.

4.7 | Equivalence of R, r, and ε^{-1}

In this section we will prove the equivalence of the three parameters R, r and ε^{-1} for SDP-solving. That is, we will show any two of the three parameters can be made constant by increasing the third. Therefore, $\frac{Rr}{\varepsilon}$ as a whole is the interesting parameter. This appendix is structured as a set of reductions, in each case we will denote the parameters of the new SDP with a tilde.

Lemma 4.35 *For every SDP with $R, r \geq 1$ and $0 < \varepsilon \leq 1$, there is an SDP with parameters $\tilde{R} = 1$ and $\tilde{r} = r$ such that solving that SDP with precision $\tilde{\varepsilon} = \frac{\varepsilon}{R}$ solves the original SDP.*

Proof. Let $\tilde{A}_j = A_j$, $\tilde{C} = C$ and $\tilde{b} = \frac{b}{R}$. Now clearly $\tilde{R} = 1$, but $\widetilde{\mathrm{OPT}} = \mathrm{OPT}/R$. Hence determining $\widetilde{\mathrm{OPT}}$ up to additive error $\tilde{\varepsilon} = \frac{\varepsilon}{R}$ will determine OPT up to additive error ε. Notice that the feasible region of the dual did not change, so $\tilde{r} = r$. □

Lemma 4.36 *For every SDP with $R, r \geq 1$ and $0 < \varepsilon \leq 1$, there is an SDP with parameters $\tilde{R} = \frac{R}{\varepsilon}$ and $\tilde{r} = r$ such that solving that SDP with precision $\tilde{\varepsilon} = 1$ solves the original SDP.*

Proof. Let $\tilde{A}_j = A_j$, $\tilde{C} = C$ and $\tilde{b} = \frac{b}{\varepsilon}$. Now $\tilde{R} = \frac{R}{\varepsilon}$ and $\widetilde{\mathrm{OPT}} = \mathrm{OPT}/\varepsilon$. Hence determining $\widetilde{\mathrm{OPT}}$ up to additive error $\tilde{\varepsilon} = 1$ will determine OPT up to additive error ε. Notice that again the feasible region of the dual did not change, so $\tilde{r} = r$. □

Lemma 4.37 *For every SDP with $R, r \geq 1$ and $0 < \varepsilon \leq 1$, there is an SDP with parameters $\tilde{R} = R$ and $\tilde{r} = 1$ such that solving that SDP with precision $\tilde{\varepsilon} = \frac{\varepsilon}{r}$ solves the original SDP.*

Proof. Let $\tilde{A}_j = A_j$, $\tilde{b} = b$ and $\tilde{C} = \frac{1}{r}C$. Now $\tilde{r} = 1$ and $\widetilde{\mathrm{OPT}} = \mathrm{OPT}/r$. Hence determining $\widetilde{\mathrm{OPT}}$ up to additive error $\tilde{\varepsilon} = \frac{\varepsilon}{r}$ will determine OPT up to additive error ε. Since $r \geq 1$ and $\|C\| \leq 1$, we find $\|\tilde{C}\| \leq 1$ as required. Notice that the feasible region of the primal did not change, so $\tilde{R} = R$. □

At this point we would like to state the final reduction, the reduction that increases r in order to get a constant ε, by setting $\tilde{C} = \frac{1}{\varepsilon}C$, but this would not guarantee that $\|\tilde{C}\| \leq 1$. Instead we show that each step of the binary search for the optimum can be performed with constant error by increasing r.

Lemma 4.38 *Given an SDP with $R, r \geq 1$, a precision $0 < \varepsilon \leq 1$ and a guess for the optimum ω such that $|\omega - \mathrm{OPT}| \leq 2\varepsilon$, there is an SDP with parameters $\tilde{R} = \mathcal{O}(R)$ and $\tilde{r} = \mathcal{O}(r/\varepsilon)$ such that solving that SDP with precision $\tilde{\varepsilon} = 1/2$ yields a new guess ω' for the optimum of the original SDP with $|\omega' - \mathrm{OPT}| \leq \frac{3}{2}\varepsilon$.*

Proof. We introduce a new variable z and consider the SDP

$$\begin{aligned} \max_{z\in\mathbb{R}, X\in\mathbb{R}^{n\times n}} \quad & z \\ \text{s.t.} \quad & \varepsilon z - \mathrm{Tr}(CX) \leq -\omega + 2\varepsilon \\ & \mathrm{Tr}\big(A_j X\big) \leq b_j \quad \text{for all } j \in [m] \\ & z \geq 0, X \succeq 0. \end{aligned}$$

Rewriting the first constraint gives $z \leq \frac{\mathrm{Tr}(CX)-\omega}{\varepsilon} + 2$. Hence z can be maximized by letting X be an optimal solution to the original SDP. It follows that $\widetilde{\mathrm{OPT}} = \frac{\mathrm{OPT}-\omega}{\varepsilon} + 2 \in [0,4]$. Hence we can let $\tilde{R} = R + 4$. Furthermore, learning whether $\widetilde{\mathrm{OPT}} \leq 5/2$ or $\widetilde{\mathrm{OPT}} \geq 3/2$ yields a new guess for OPT with error less than $\frac{3}{2}\varepsilon$.

It remains to give an upper bound on $\tilde{r}$. In order to do this, consider the dual program

$$\begin{aligned} \min_{v\in\mathbb{R}, y\in\mathbb{R}^m} \quad & \langle b, y\rangle + (-\omega + 2\varepsilon) v \\ \text{s.t.} \quad & \varepsilon v \geq 1 \\ & \sum_{j=1}^m y_j A_j - vC \succeq 0 \\ & y \geq 0, v \geq 0. \end{aligned}$$

Note that the constraints imply $v \geq 1/\varepsilon$. We claim that this is tight in the optimum. To see this, fix v to some value $v_0 \geq 1/\varepsilon$ and consider the SDP in the remaining variables. Let y^* be an optimizer for the original dual SDP with ℓ_1-norm equal to r. Then clearly $v_0 y^*$ is an optimizer for this new dual SDP with v fixed. The objective value of this optimizer is $v_0\big(\langle b, y^*\rangle - \omega + 2\varepsilon\big)$. However, by assumption

$$\langle b, y^*\rangle - \omega + 2\varepsilon = \mathrm{OPT} - \omega + 2\varepsilon \geq -2\varepsilon + 2\varepsilon = 0.$$

It follows that the optimal value can only increase by increasing v above $1/\varepsilon$, and hence $v = 1/\varepsilon$ and $y = y^*/\varepsilon$ forms an optimizer for the new SDP. This shows that $\tilde{r} = r/\varepsilon + 1/\varepsilon = \mathcal{O}(r/\varepsilon)$ suffices. □

4.8 | Applications

Although quantum SDP-solvers provide a speed up in terms of n and m over classical algorithms they do suffer an increased dependence on R, r and ε. In the first two papers on quantum SDP-solving [BS17; AGGW17] it remained an open question whether any applications could be found in the regime where Rr/ε was small enough to get a speedup over the best classical methods. Later Brandão et al. [BKLLSW19] showed that SDP primal oracles can be used to solve the problem of shadow tomography if the input is given in the quantum state input model. Shadow tomography was recently proposed by Aaronson [Aar18], who also gave a sample-efficient algorithm. In Section 4.8.1 we apply our improved SDP primal oracles to this problem, simultaneously improving the sample complexity and computational complexity compared to the previous works.

We also propose new applications to quantum SDP-solvers, namely the problems of *quantum state discrimination* and *E-optimal design*. In both cases we show a speedup over the best possible classical algorithm in terms of some input parameters, at the expense of a large dependence on other parameters. Therefore we do not believe these results to be useful in practice. However, they do show that an improvement in some parameters is possible, and further research in this area might be able to improve the error dependence for these special cases. We end this section with a short discussion on how out SDP-solvers can be used to implement LP-solvers.

4.8.1 | Improved shadow tomography

We apply the idea from Brandão et al. [BKLLSW19] to use an SDP primal oracle to the problem of *shadow tomography* proposed by Aaronson [Aar18]. In *shadow tomography* we are given the ability to prepare samples of an n-dimensional quantum state τ and we have a description of some measurement operators $E_1,\ldots,E_m$; the goal is to find ε-approximations of the corresponding expectation values $\mathrm{Tr}\big(E_j\tau\big)$ for all $j\in[m]$. Aaronson showed that this can be done with only[20]

$$\widetilde{\mathcal{O}}\left(\frac{\log(m)^4\log(n)}{\varepsilon^5}\right)$$

samples from τ, but his method has high computational costs. Note that here there are three different complexity measures: the number of samples from τ, the number of queries to some input model that describes the E_j, and the number of elementary operations.

In [BKLLSW19] Brandão et al. showed that a slightly relaxed problem can be efficiently solved using an SDP primal oracle. The problem they solved is to find a $y\in\mathbb{R}^m$ for which $\sigma := e^{-\sum_{j=1}^m y_j E_j}/\mathrm{Tr}\Big(e^{-\sum_{j=1}^m y_j E_j}\Big)$ is such that $|\mathrm{Tr}\big(E_j(\tau-\sigma)\big)|\le\varepsilon/2\ \forall j\in[m]$, i.e.,

[20] Aaronson improves the ε-dependence of his algorithm to match our upper bound from Theorem 4.39 in a version of [Aar18] that appeared after our work.

σ is in

$$\begin{aligned}\mathscr{P}_\varepsilon = \{\sigma : \ & \sigma \succeq 0,\\ & \mathrm{Tr}(\sigma) = 1,\\ & \mathrm{Tr}\left(\sigma E_j\right) \leq \mathrm{Tr}\left(\tau E_j\right) + \varepsilon/2 \quad \forall j \in [m],\\ & \mathrm{Tr}\left(-\sigma E_j\right) \leq \mathrm{Tr}\left(-\tau E_j\right) - \varepsilon/2 \quad \forall j \in [m]\}.\end{aligned}$$

We call the problem of finding a classical description of τ that suffices to solve the shadow tomography problem without any more samples from τ the *descriptive shadow tomography problem.* In particular if we get a vector y as above, then for a given $j \in [m]$ using $\widetilde{\mathcal{O}}_m\left(1/\varepsilon^2\right)$ invocations of a Gibbs-sampler for y followed by the measurement E_j suffices to find an ε-approximation of $\mathrm{Tr}\left(\tau E_j\right)$ with success probability at least $1 - \mathcal{O}(1/m)$ without any more samples from τ. If we can coherently apply E_j, then using amplitude estimation techniques the number of (coherent) Gibbs-sampler calls can be reduced to $\widetilde{\mathcal{O}}_m(1/\varepsilon)$.

Due to the output size of the shadow tomography problem, a trivial $\Omega\left(m\log(1/\varepsilon)\right)$ lower bound can be given on the gate complexity, since we need to write down the results. However, this limitation does not exist for the descriptive shadow tomography problem. Both problems clearly have the same sample complexity, furthermore the best known lower bound on the sample complexity is $\Omega\left(\log(m)/\varepsilon^2\right)$ [Aar18].

Theorem 4.39 *The descriptive shadow tomography problem can be solved using*

$$\widetilde{\mathcal{O}}\left(\frac{\log(m)^4\log(n)}{\varepsilon^4}\right)$$

samples from τ. Furthermore, when the E_j matrices are accessible in the quantum operator input model, then this can be done using

$$\widetilde{\mathcal{O}}\left(\left(\sqrt{m} + \frac{\sqrt{n}}{\varepsilon}\right)\frac{\alpha}{\varepsilon^4}\right)$$

queries to the E_j and elementary operations. It follows that the same bound holds with $\alpha = s$ for the sparse matrix input model.

Proof. The samples from τ are only used for calculating the values b_j, i.e., calculating $\frac{\varepsilon}{4}$-approximations of $\mathrm{Tr}\left(\tau E_j\right)$, when checking the constraints in the SDP primal oracle. Like in [BKLLSW19] we make a small adjustment to our SDP primal oracle: when Gibbs-sampling the Gibbs state ρ, we also sample τ to create the state $\rho \otimes \tau$. Then, when checking the constraint, we measure $E_j \otimes I - I \otimes E_j$ to obtain an approximation of $\mathrm{Tr}\left(E_j\rho\right) - \mathrm{Tr}\left(E_j\tau\right)$. Notice that our SDP primal oracle uses $\widetilde{\mathcal{O}}\left(\frac{\log(m)^4\log(n)}{\varepsilon^4}\right)$ Gibbs states, $\widetilde{\mathcal{O}}_{\log(n)}\left(\log(m)^4/\varepsilon^2\right)$ in each of the $\mathcal{O}\left(\log(n)/\varepsilon^2\right)$ iterations, and hence the modified version uses that many samples from τ too.

The statements about the query complexity and number of elementary operations follow directly from Theorem 4.34. □

As a final remark we note that the POVM input model we introduced in Section 4.5.1 is very natural in this situation. In particular, this shows that $\widetilde{\mathcal{O}}\big((\sqrt{m}+\sqrt{n}/\varepsilon)(1/\varepsilon)^4\big)$ applications of the measurements suffice.

4.8.2 | Quantum state discrimination

In the *Quantum State Discrimination* problem we are given k d-dimensional quantum states $\rho^{(1)},\dots,\rho^{(k)} \in \mathbb{C}^{d\times d}$, in some input model. Our goal is to find a POVM $M^{(1)},\dots,M^{(k)}$ that has the "best" probability of discriminating between the states. Here "best" has two natural meanings:

- The minimal success probability is maximized: $\max_M \min_{i\in[k]} \operatorname{Tr}\big(M^{(i)}\rho^{(i)}\big)$.
- The sum of all the success probabilities is maximized.

Both problems can be cast as an SDP [Eld03] but we will only consider the second here since it lends it self better to the Arora-Kale framework. For the rest of the section, when we refer to the quantum state discrimination problem, we mean the problem of maximizing the sum of success probabilities. Our goal will be to get a quantum speedup in k at the expense of a slowdown in terms of d. Fortunately, the interesting cases of the problem seem to occur when $d \ll k$. Furthermore, we will use the quantum state input model as introduced in Section 4.5.1 to show that it is possible to solve the problem even when only given access to unitaries that prepare the quantum states.

Theorem 4.40 *Given access to the matrix entries of the quantum states $\rho^{(1)},\dots,\rho^{(k)} \in \mathbb{C}^{d\times d}$ the quantum state discrimination problem can be solved up to additive error ε on a quantum computer using*

$$\widetilde{\mathcal{O}}\left(\frac{\sqrt{k}}{\varepsilon^5}\operatorname{poly}(d)\right)$$

queries to the input.

Given access to a unitary that creates a purified version of the quantum states $\rho^{(1)},\dots,\rho^{(k)} \in \mathbb{C}^{d\times d}$ the quantum state discrimination problem can be solved up to additive error ε on a quantum computer using

$$\widetilde{\mathcal{O}}\left(\frac{k^{1.5}}{\varepsilon^5}\operatorname{poly}(d)\right)$$

queries.

Proof. To maximize the sum of success probabilities, notice that the probability of identifying $\rho^{(i)}$ correctly is $\mathrm{Tr}\big(M^{(i)}\rho^{(i)}\big)$. Writing the problem as an SDP we get:

$$\begin{aligned}\max\quad & \sum_{i=1}^{k}\mathrm{Tr}\Big(M^{(i)}\rho^{(i)}\Big)\\ \text{s.t.}\quad & \sum_{i=1}^{k} M^{(i)} = I_d\\ & M^{(i)} \succeq 0 \text{ for all } i\in[k].\end{aligned}$$

This can be written in the standard form (4.1) as follows:

- $X = \mathrm{diag}\big(M^{(1)},\dots,M^{(k)}\big)$.
- $C = \mathrm{diag}\big(\rho^{(1)},\dots,\rho^{(k)}\big)$.
- For $s,t\in[d]$ we let $A_{st} = \oplus_{i=1}^{k} E_{st}$ and $b_{st} = \delta_{st}$, where δ_{st} is the Kronecker delta.

Notice that we have strict equalities, as opposed to the inequalities in the standard form. These equalities can be cast into inequality form by adding a separate upper and lower bound. However, this is not needed for the analysis, just note that an equality in the primal corresponds to a variable in the dual without positivity constraint.

To apply our SDP-solvers we need to give bounds on the input parameters. Clearly here $n = kd$ and $m = \mathcal{O}\big(d^2\big)$. Furthermore, since the objective matrix is block diagonal with $d\times d$ blocks, the sparsity s is at most d. To bound the parameter B in the quantum state input model, note that C has trace k and is psd, and all other constraints can clearly be decomposed with a constant trace. It remains to give a bound for R and r.

For R, the bound on the trace of a primal solution, notice that

$$\mathrm{Tr}(X) = \sum_{i=1}^{k}\mathrm{Tr}\Big(M^{(i)}\Big) = \mathrm{Tr}\left(\sum_{i=1}^{k} M^{(i)}\right) = \mathrm{Tr}(I_d) = d.$$

For r we need to write out the dual. Doing so directly gives:

$$\begin{aligned}\min\quad & \sum_{s,t\in[d]} y_{st}\delta_{st} = \sum_{s\in[d]} y_{ss}\\ \text{s.t.}\quad & \sum_{s,t\in[d]} y_{st}\Big(\oplus_{i=1}^{k}E_{st}\Big) \succeq \oplus_{i=1}^{k}\rho^{(i)}.\end{aligned}$$

Notice that we do not have a $y\geq 0$ constraint since we have strict equalities in the primal. We could have replaced the equalities by inequalities and then we would get a y^+ and y^- vector, both non-negative, such that $y = y^+ - y^-$. However, since r is a upper bound on the sum of the values in *one* optimal solution, it is enough to bound the absolute value of the y variables.

To do so, simply rewrite the dual with a separate constraint for each $\rho^{(i)}$ and reorganize the y_{st} variables in a $d\times d$ matrix Y:

$$\begin{aligned}\min\quad & \mathrm{Tr}(Y)\\ \text{s.t.}\quad & Y \succeq \rho^{(i)} \text{ for all } i\in[k].\end{aligned}$$

Clearly I_d is feasible for this problem, so for an optimal Y we have $\mathrm{OPT} = \mathrm{Tr}(Y) \le d$. This gives the following bound, for $S \in \{-1,1\}^{d\times d}$,

$$\sum_{s,t}^{d} |Y_{st}| = \max_{S\in\{-1,1\}^{d\times d}} \mathrm{Tr}(SY) \le \mathrm{Tr}(Y) \max_{S\in\{-1,1\}^{d\times d}} \|S\| \le d\mathrm{Tr}(Y) \le d^2, \tag{4.10}$$

so $r = d^2$ suffices.[21]

Applying our results about the complexity of SDP-solving gives the claimed bounds.

□

Applying our SDP-solver results in a dual solution Y and a description of a primal solution X. This description of the primal solution is of the form (Y', y_0) and describes to primal solution via $M^{(i)} \propto e^{Y'-y_0\rho^i}$. Note that this representation gives an interesting way of compressing a POVM, since the Y' matrix has dimension $d \times d$, but encodes k POVM operators with the help of the $\rho^{(i)}$ matrices. The dual solution Y could be of independent interest too, solving the following problem: for a set of density operators $\rho^{(i)}$, find the matrix Y with the smallest trace such that $Y \succeq \rho^{(i)}$ for all i.

A lower bound To find a lower bound, fix $d = 2$, i.e., consider a single qubit. Now let $z \in \{0,1\}^k$ be the input for a search problem. We want to distinguish the cases $|z| = 0$ and $|z| = 1$ under the promise that we are in one of these cases. This is known to take $\Omega\left(\sqrt{k}\right)$ quantum queries or $\Omega(k)$ classical queries, see also Chapter 6. Now let $\rho^{(j)} = |z_j\rangle\langle z_j|$. Given query access to z it is easy to construct the input oracles for the different input models. Clearly if $z = 0^k$, then all states are equal, thus the sum of success probabilities is always 1. However, if $z_k = 1$, then by setting $M^{(k)} := |1\rangle\langle 1|$ and choosing the other measurement operators arbitrarily, we clearly get a sum of success probabilities of 2. Hence a 1/3-approximation to the optimal value of the SDP given above will solve the search problem and hence takes at least $\Omega\left(\sqrt{k}\right)$ quantum queries or at least $\Omega(k)$ classical queries.

4.8.3 | Optimal design

In the optimal design setting we want to learn a hidden (not necessarily quantum) state $\theta \in \mathbb{R}^d$ through experiments. There is a set of k possible experiments, represented by unit vectors $u^{(1)},\ldots,u^{(k)} \in \mathbb{R}^d$, and when we execute the ith experiment we learn $\langle\theta, u^{(j)}\rangle$ with some Gaussian noise. In particular we get a sample from $\mathcal{N}(\langle\theta, u^{(j)}\rangle, \sigma_j)$. Precise estimation of θ requires a lot of experiments, and the problem in optimal design is to decide which distribution to use when choosing the experiments in order to "minimize"

[21] This also proves that for k states of dimension d, the sum of success probabilities of discrimination is always at most d, so the average will be at most d/k. Thus the error parameter should scale with $1/k$ if we would consider the average probability. This is why we choose to look at the sum of success probabilities instead.

the covariance matrix of the maximum likelihood estimator of θ. Since the variance of the maximum likelihood estimator is hard to express analytically, we instead look at the *Fischer information matrix*, which is a good approximation for the inverse of the covariance matrix, and has a nice closed form:

$$F_p = \sum_{i=1}^{k} p_i u^{(i)} u^{(i)T} / \sigma_i^2,$$

where p_i is the probability of doing experiment $u^{(i)}$. Now, to make the covariance matrix "small" we would like to make the Fischer information matrix "large". For a more detailed explanation, see for example [Sil80].

The precise meaning of "small" and "large" can be defined in several sensible ways. The most common criteria are called *A-optimal*, *D-optimal* and *E-optimal design*. In A-optimal design we want to minimize the sum of the eigenvalues of the covariance matrix, or as an approximation the sum of the eigenvalues of the inverse of the Fischer information matrix. Unfortunately the SDP formulation of this problem has parameters r, R that make our methods inefficient. In D-optimal design we want to minimize the determinant of the covariance matrix, this can be approximated with a convex program, but sadly this problem does not naturally correspond to an SDP.

We will consider E-optimal design. In this setting we would like to minimize the operator norm of the covariance matrix. Since this is hard to do, we will try to maximize the smallest eigenvalue of the Fischer information matrix. Let $P := \max_i \frac{1}{d\sigma_i^2}$ be an input parameter dependent on the maximal variance of the experiments. Note that if the size d of the states increases but the experiments still have the same variance, then the parameter P decreases. In fact, P can be viewed as a measure of how much information a single measurement can give about each of the coordinates.

Theorem 4.41 *The E-optimal design problem, that is, finding a distribution $p \in \Delta^k$ such that the smallest eigenvalue of F_p is maximized, can be solved up to additive error ε using sparse access to the s-sparse experiment (unit) vectors $u^{(1)}, \ldots, u^{(k)} \in \mathbb{R}^d$ and oracle access to the σ_i values with*

$$\widetilde{\mathcal{O}}\left(\left(\sqrt{k} + \sqrt{d}\frac{P^2}{\varepsilon}\right) s \frac{P^8}{\varepsilon^4}\right)$$

queries on a quantum computer.

Proof. We consider the following SDP:

$$\begin{aligned} \max \quad & t \\ \text{s.t.} \quad & \sum_{i=1}^{k} p_i u^{(i)} u^{(i)T} / \sigma_i^2 \succeq t I_d \\ & \sum_{i=1}^{k} p_i \leq 1 \\ & p_i \geq 0 \text{ for all } i \in [k]. \end{aligned}$$

Clearly this SDP would maximize the minimal eigenvalue of $F_p = \sum_{i=1}^k p_i u^{(i)} u^{(i)T}/\sigma_i^2$. We can rewrite this in standard dual form, flipping the sign of the optimal value:

$$\begin{aligned} \min \quad & -t \\ \text{s.t.} \quad & \sum_{i=1}^k p_i \begin{bmatrix} -1 & \\ & u^{(i)} u^{(i)T}/\sigma_i^2 \end{bmatrix} + t \begin{bmatrix} 0 & \\ & -I_d \end{bmatrix} \succeq \begin{bmatrix} -1 & \\ & 0 \end{bmatrix} \\ & p_i \geq 0 \text{ for all } i \in [k] \\ & t \geq 0. \end{aligned}$$

The corresponding primal problem is then easy to write down:

$$\begin{aligned} \max \quad & -z \\ \text{s.t.} \quad & -z + \operatorname{Tr}\left(X u^{(i)} u^{(i)T}\right)/\sigma_i^2 \leq 0 \text{ for all } i \in [k] \\ & \operatorname{Tr}(X) \geq 1 \\ & z \geq 0, X \succeq 0. \end{aligned}$$

From the size of the input it follows that $n = 1 + d$ and $m = 1 + k$ for this SDP. Furthermore, the row sparsity of the constraint matrices is equal to the vector sparsity of the $u^{(i)}$, which justifies the use of s for the sparsity of the vectors $u^{(i)}$. It remains to give a bound on r and R. Note that the trace constraint on X will be tight for an optimal X and hence $R = 1 + |\text{OPT}|$, where OPT is the optimal value is the optimal value of the last SDP. Similar for the sum constraint on p, we get $r = 1 + |\text{OPT}|$. To give a bound on $|\text{OPT}|$, we rewrite the primal again, flip the sign of the optimum:

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & \operatorname{Tr}\left(X u^{(i)} u^{(i)T}\right)/\sigma_i^2 \leq z \text{ for all } i \in [k] \\ & \operatorname{Tr}(X) \geq 1 \\ & z \geq 0, X \succeq 0. \end{aligned}$$

Now, let us construct a feasible point. Since we have a minimization SDP, this will give an upper bound on OPT, which is also an upper bound on |OPT| due to the non-negativity of z. Let $X = I_d/d$, then $\operatorname{Tr}\left(X u^{(i)} u^{(i)T}\right) = 1/d$, so picking $z = \max_i \frac{1}{d\sigma_i^2}$ will give a feasible point with objective value z. We conclude that $r = R = \mathcal{O}\left(\max_i \frac{1}{d\sigma_i^2}\right) = P$ suffices. The stated complexity follows using our complexity bounds on quantum SDP-solving. □

4.8.4 | Linear programming

Since semidefinite programming is an extension of linear programming, LP-solving is a natural application for our SDP-solvers. Remember that linear programs correspond to SDPs where all input matrices are diagonal. This means that the SDP sparsity s will always be 1. Access to the entries of the constraint matrix of an LP via an oracle acting as $O|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|A_{ij}\rangle$ suffices to implement the sparse matrix input model for SDPs.

The constant r has the same meaning for LPs: a bound on the ℓ_1-norm of a dual optimizer. Similar, the bound R corresponds with a bound on the ℓ_1-norm of a primal solution. Naturally we get a quantum LP-solver that uses

$$\widetilde{\mathcal{O}}((\sqrt{m} + \sqrt{n}\gamma)\gamma^4)$$

queries and elementary operations.

However, this leaves open two questions. First, is there a better algorithm possible by considering the extra structure of LPs? And second, the constraint matrix of the LP might be a sparse matrix, but our SDP algorithm does not consider this. Can we further speed up LP-solving by considering the sparsity of the LP constraints? We answer both of these questions affirmatively in the next chapter.

Chapter 5

Quantum algorithms for zero-sum games

We derive sublinear-time quantum algorithms for approximating Nash equilibria of two-player zero-sum games, based on efficient Gibbs-sampling methods. We are able to achieve quantum speedups over the best classical algorithms for both dense and sparse payoff matrices at the cost of a mildly increased dependence on the additive error. In particular we can find ε-approximate Nash equilibrium strategies in complexity $\widetilde{\mathcal{O}}\big(\sqrt{n+m}/\varepsilon^3\big)$ and $\widetilde{\mathcal{O}}_{nm}\big(\sqrt{s}/\varepsilon^{3.5}\big)$ respectively, where $n \times m$ is the size of the matrix describing the game, and s is the maximal row and column sparsity. Our algorithms use the LP formulation of the problem and apply the techniques developed in Chapter 4. We also show how to reduce general LP-solving to zero-sum games, resulting in quantum LP-solvers that have complexities $\widetilde{\mathcal{O}}\big(\sqrt{n+m}\gamma^3\big)$ and $\widetilde{\mathcal{O}}_{nm}\big(\sqrt{s}\gamma^{3.5}\big)$ for the dense and sparse access models respectively, where γ is the relevant "scale-invariant" precision parameter.

This chapter is based on the paper *Quantum algorithms for zero-sum games* by J. van Apeldoorn and A. Gilyén [AG19b].

5.1 | Introduction

A *matrix game* is a two-player zero-sum game in which both players have only a finite number of pure strategies. We label the moves for the first player (called Alice) by $[n]$ and the moves for the second player (called Bob) by $[m]$. If Alice plays $i \in [n]$ and Bob plays $j \in [m]$, then Alice has to pay Bob $A_{ji} \in [-1,1]$. We say that Alice receives a *payoff* $-A_{ji}$ and Bob receives a payoff A_{ji}. Their individual goal is to get the highest payoff possible. The payoffs can be written in matrix form $A \in [-1,1]^{m\times n}$, hence the name *matrix game*. A game is called *symmetric* if $m = n$ and $A = -A^T$, in other words, if the payoff matrix is *skew symmetric*.

If one of the players would always play the same move, then (for most games) it would be easy for the other player to get a large payoff. Hence a strategy will be randomized in general. As in Section 2.4.1, let Δ^n be the set of all non-negative vectors in $\mathbb{R}^n$ that sum to 1, i.e., the set of all probability distributions over n elements. Given a randomized strategy $x \in \Delta^n$ for Alice and a randomized strategy $y \in \Delta^m$ for Bob, the expected payoff for Bob is $y^T Ax$.

Naturally Alice's goal is to minimize Bob's expected payoff; the best she can do is to assume that Bob plays the best strategy y on his side and optimize her x for that:

$$\min_{x\in\Delta^n}\max_{y\in\Delta^m} y^T Ax.$$

We can write this as a linear program (LP) by noting that a linear function over the simplex (in this case $y \mapsto y^T Ax$) is maximized on a vertex of the simplex:

$$\min_{x\in\Delta^n}\max_{y\in\Delta^m} y^T Ax = \min_{x\in\Delta^n}\max_{j\in[m]} e_j^T Ax,$$

which can be written as an LP:

$$\begin{aligned} \min\quad & \lambda \\ s.t.\quad & Ax \leq \lambda\mathbf{1} \\ & x \in \Delta^n \\ & \lambda \in \mathbb{R}. \end{aligned} \tag{5.1}$$

Notice that since Bob's strategy y is in Δ^m he can indeed not get a better expected value than λ. In fact, the *dual* of this LP is the LP that Bob would get if he reasoned similarly. Due to strong duality both LPs give the same value! Hence we will call the optimal λ^* in (5.1) the *value* of the game. The corresponding strategies are together called a *Nash equilibrium*. Notice that for a symmetric game the value is always 0 since no player can have an advantage, and that in general $\lambda^* \in [-1,1]$. For completeness, let us also state the dual problem:

$$\begin{aligned} \max\quad & \lambda' \\ s.t.\quad & A^T y \geq \lambda'\mathbf{1} \\ & y \in \Delta^n \\ & \lambda' \in \mathbb{R}. \end{aligned} \tag{5.2}$$

We will call a strategy x for Alice *ε-optimal* if $Ax \leq (\lambda^* + \varepsilon)\mathbf{1}$, and similarly for Bob. Grigoriadis and Khachiyan [GK95] showed that a pair of ε-optimal strategies can be found using a classical computation consisting of $\mathcal{O}\big(\log(n+m)/\varepsilon^2\big)$ iterations, and with $\mathcal{O}\big((n+m)/\log(n+m)\big)$ steps per iteration.[1] Notice that this leads to a *sub-linear* amount of work! They show this by first converting any game to a symmetric game and then showing that symmetric games can be solved by a randomized algorithm. In Section 5.2, we give a proof of their results that directly applies to non-symmetric games. In fact, a more careful analysis shows that an iteration can be implemented in $\widetilde{\mathcal{O}}_{\frac{mn}{\varepsilon}}(s)$ operations, where s is the maximal row and column sparsity of A. In the same paper the authors also prove that any *deterministic* algorithm would require at least $mn/2$ queries to the input.

As in Chapter 4 we will show that using a quantum computer a quadratic improvement in terms of the dimensions n and m can be achieved, at the expense of a slightly heavier dependence on the precision. In particular, in Section 5.3.1 we show that on a quantum computer $\widetilde{\mathcal{O}}\big(\sqrt{n+m}/\varepsilon^3\big)$ queries to the entries of A and elementary operations suffice to implement the algorithm by Grigoriadis and Khachiyan [GK95]. In Section 5.3.2 we show that this can be improved to $\widetilde{\mathcal{O}}_{nm}\big(\sqrt{s}/\varepsilon^{3.5}\big)$ for sparse matrices. Note that unlike the algorithms for SDP-solving, the classical algorithm that we speed up in this chapter (and hence our quantum algorithm) does not depend on additional scale parameters, thus the achieved speedups seem more applicable in practice.

In Section 5.4 we also show how to reduce general LP-solving to zero-sum games, resulting in new general-purpose quantum LP-solvers. This reduction reintroduces the extra dependence on the size of the primal and dual solutions into the complexity. However, the dependence on these parameters, and on the approximation error, is only cubic, whereas an LP-solver obtained from our results on SDP-solving from Chapter 4 would have a fifth power dependence. Furthermore, we give the first quantum LP-solver which depends on the sparsity of the LP instead of on n and m.[2]

Computational model We will work in the same computational model as we did in Chapter 4. This means that we will again assume access to a classical-write / quantum-read random access memory as an elementary operation. We only require such a memory consisting of $\widetilde{\mathcal{O}}_{nm}\big(1/\varepsilon^2\big)$ bits, so not allowing such an operation will worsen the gate complexity by at most a factor of $\widetilde{\mathcal{O}}_{nm}\big(1/\varepsilon^2\big)$.

5.2 | Classical algorithm

In this section we will present the classical zero-sum game algorithm developed by Grigoriadis and Khachiyan [GK95], with two alterations:

[1] The authors show that the steps in each iteration can be highly parallelized.

[2] The sparsity of an LP should not be confused with the sparsity parameter relevant in SDP solving. An LP that is written as an SDP will have SDP sparsity 1 since all the matrices involved are diagonal. Our goal here is to get a dependence on the LP sparsity instead of a dependence on n and m.

1. We give the algorithm and its proof without first reducing the problem to symmetric games. In this way we lay more emphasis on the fact that this is a *primal-dual* approach and on the connection to *fictitious play*. Furthermore, we hope that this view on the algorithm will allow for an easier application to other problems.

2. The algorithm by Grigoriadis and Khachiyan assumes the desired additive error ε is known from the start of the algorithm. We present a version of the algorithm for which the additive error in the intermediate solutions uniformly decreases during the run of the algorithm, but we also consider a version corresponding to a fixed accuracy goal more similar to their results.

```
$x^{(1)} \leftarrow 0 \in \mathbb{R}^n$ and $y^{(1)} \leftarrow 0 \in \mathbb{R}^m$.
for $t = 1, 2, \ldots$ do
    $\eta^{(t)} := \frac{1}{2\sqrt{t}}$ (alternatively in the fixed accuracy-goal version choose $\eta^{(t)} := \varepsilon/4$).
    $u^{(t)} := Ax^{(t)}$ and $v^{(t)} := -A^T y^{(t)}$.
    $P^{(t)} := e^{u^{(t)}} = e^{Ax^{(t)}}$ and $Q^{(t)} := e^{v^{(t)}} = e^{-A^T y^{(t)}}$.
    $p^{(t)} := P^{(t)}/\|P^{(t)}\|_1$ and $q^{(t)} := Q^{(t)}/\|Q^{(t)}\|_1$.
    Sample $a \sim p^{(t)}$ and $b \sim q^{(t)}$.
    $y^{(t+1)} = y^{(t)} + \eta^{(t)} e_a$ and $x^{(t+1)} = x^{(t)} + \eta^{(t)} e_b$.
end for
```

Algorithm 5.1: Main Zero-Sum game algorithm

We start by proving that this algorithm is correct, before giving an upper bound on the complexity.

Lemma 5.1 *Let $\delta \in (0, 1/3)$. With probability at least $1-\delta$ Algorithm 5.1 produces a sequence of $x^{(t)}$ and $y^{(t)}$ such that for all t the intermediate solutions $x^{(t)}/\|x^{(t)}\|_1$ and $y^{(t)}/\|y^{(t)}\|_1$ are ε'-optimal solutions with*

$$\varepsilon' = \frac{2}{\sqrt{t}} \cdot (3\ln(t) + \ln(nm) + \ln(1/\delta) + 2).$$

Let $\varepsilon \in (0,1)$. If we run the algorithm with $\eta^{(t)} := \varepsilon/4$ instead, then the solutions are ε-optimal with probability at least $1-\delta$ after $T = \left\lceil \frac{16\ln\left(\frac{nm}{\delta}\right)}{\varepsilon^2} \right\rceil$ iterations.

Proof. We start by noting that $x^{(t)}/\|x^{(t)}\|_1$ and $y^{(t)}/\|y^{(t)}\|_1$ are indeed probability distributions. Hence $\left(x^{(t)}/\|x^{(t)}\|_1, \max_j (Ax^{(t)}/\|x^{(t)}\|_1)_j\right) \in \mathbb{R}^{n+1}$ is a feasible point for the primal (5.1) and $\left(y^{(t)}/\|y^{(t)}\|_1, \min_i (A^T y^{(t)}/\|y^{(t)}\|_1)_i\right) \in \mathbb{R}^{m+1}$ is a feasible point for the dual (5.2). Due to strong duality, to show that these solutions are ε-optimal, it suffices to show that the difference between their values is at most ε. We do so by showing that

$$\forall i \in [n], j \in [m] : \left(Ax^{(t)}/\|x^{(t)}\|_1\right)_j - \left(A^T y^{(t)}/\|y^{(t)}\|_1\right)_i \leq \varepsilon,$$

since this implies that $\max_j (Ax^{(t)}/\|x^{(t)}\|_1)_j - \min_i (A^T y^{(t)}/\|y^{(t)}\|_1)_i \leq \varepsilon$.

To do so we consider the potential function

$$\Phi(t) := \left(\sum_{j\in[m]} P_j^{(t)}\right)\left(\sum_{i\in[n]} Q_i^{(t)}\right) = \left(\sum_{j\in[m],i\in[n]} P_j^{(t)} Q_i^{(t)}\right)$$

and show that this is bounded from above. In the beginning $\Phi(1) = nm$, moreover

$$\begin{aligned}
\Phi(t+1) &= \left(\sum_{j\in[m]} P_j^{(t+1)}\right)\left(\sum_{i\in[n]} Q_i^{(t+1)}\right)\\
&= \left(\sum_{j\in[m]} e^{\left(Ax^{(t)}\right)_j + \eta^{(t)} A_{jb}}\right)\left(\sum_{i\in[n]} e^{-\left(A^T y^{(t)}\right)_i - \eta^{(t)} A_{ai}}\right)\\
&= \left(\sum_{j\in[m]} P_j^{(t)} e^{\eta^{(t)} A_{jb}}\right)\left(\sum_{i\in[n]} Q_j^{(t)} e^{-\eta^{(t)} A_{ai}}\right)\\
&= \left(\sum_{j\in[m]} p_j^{(t)} \|P^{(t)}\|_1 e^{\eta^{(t)} A_{jb}}\right)\left(\sum_{i\in[n]} q_i^{(t)} \|Q^{(t)}\|_1 e^{-\eta^{(t)} A_{ai}}\right)\\
&= \|P^{(t)}\|_1 \|Q^{(t)}\|_1 \left(\sum_{j\in[m]} p_j^{(t)} e^{\eta^{(t)} A_{jb}}\right)\left(\sum_{i\in[n]} q_i^{(t)} e^{-\eta^{(t)} A_{ai}}\right)\\
&= \Phi(t)\left(\sum_{j\in[m]} p_j^{(t)} e^{\eta^{(t)} A_{jb}}\right)\left(\sum_{i\in[n]} q_i^{(t)} e^{-\eta^{(t)} A_{ai}}\right).
\end{aligned}$$

Taking the expectation over the sampling of a and b, and working out the sums, we get

$$\begin{aligned}
\mathbb{E}[\Phi(t+1)] &= \Phi(t) \sum_{a\in[m]} \sum_{b\in[n]} \sum_{j\in[m]} \sum_{i\in[n]} p_a^{(t)} q_b^{(t)} p_j^{(t)} q_i^{(t)} e^{\eta^{(t)}(A_{jb} - A_{ai})}\\
&\leq \Phi(t) \sum_{a\in[m]} \sum_{b\in[n]} \sum_{j\in[m]} \sum_{i\in[n]} p_a^{(t)} q_b^{(t)} p_j^{(t)} q_i^{(t)} \left(1 + \eta^{(t)}(A_{jb} - A_{ai}) + 3\left(\eta^{(t)}\right)^2\right),
\end{aligned}$$

where we used the fact that for all $x \in [-1,1]$ it holds that $e^x \leq 1 + x + 3x^2/4$, and that $\left|A_{jb} - A_{ai}\right|^2 \leq 4$. Now also observe that

$$\sum_{a\in[m]} \sum_{b\in[n]} \sum_{j\in[m]} \sum_{i\in[n]} p_a^{(t)} q_b^{(t)} p_j^{(t)} q_i^{(t)} = 1,$$

and that all the $A_{jb} - A_{ai}$ terms cancel against another $A_{ai} - A_{jb}$ term with the same $p_a^{(t)} q_b^{(t)} p_j^{(t)} q_i^{(t)}$ coefficient. Hence

$$\mathbb{E}[\Phi(t+1)] \leq \Phi(t)\left(1 + 3\left(\eta^{(t)}\right)^2\right),$$

and by taking the expectation on both sides and expanding the recursion we get

$$\mathbb{E}[\Phi(t)] \leq \Phi(1) \prod_{\tau=1}^{t-1}\left(1 + 3\left(\eta^{(\tau)}\right)^2\right) \leq nme^{3\Sigma_{\tau=1}^{t-1}\left(\eta^{(\tau)}\right)^2}.$$

By Markov's inequality, for any $\delta^{(t)} \in (0,1/3)$, with probability at least $1-\delta^{(t)}$, we have that

$$\Phi(t) \leq \frac{nm}{\delta^{(t)}} e^{3\sum_{\tau=1}^{t-1}\left(\eta^{(\tau)}\right)^2}.$$

Since $\Phi(t)$ is the sum of positive terms, each term itself is smaller than the sum. It follows that for all $i \in [n], j \in [m]$

$$P_j^{(t)} Q_i^{(t)} = e^{\left(Ax^{(t)}\right)_j - \left(A^T y^{(t)}\right)_i} \leq \frac{nm}{\delta^{(t)}} e^{3\sum_{\tau=1}^{t-1}\left(\eta^{(\tau)}\right)^2}.$$

Taking the natural logarithm on both sides, and dividing by $\|x^{(t)}\|_1 = \|y^{(t)}\|_1 = \sum_{\tau=1}^{t}\eta^{(\tau)}$ we get that

$$\forall i \in [n], j \in [m] : \left(Ax^{(t)}/\|x^{(t)}\|_1\right)_j - \left(A^T y^{(t)}/\|y^{(t)}\|_1\right)_i \leq \frac{\ln\left(\frac{nm}{\delta^{(t)}}\right) + 3\sum_{\tau=1}^{t}\left(\eta^{(\tau)}\right)^2}{\sum_{\tau=1}^{t}\eta^{(\tau)}}. \tag{5.3}$$

Until now every step works for both choices of $\eta^{(t)}$. First we finish the proof of the infinitely running version of the algorithm, where we choose $\delta^{(t)} := \frac{\delta}{2t^2}$. Using the bounds $\sum_{\tau=1}^{t}\left(\eta^{(\tau)}\right)^2 = \frac{1}{4}\sum_{\tau=1}^{t}\frac{1}{\tau} \leq \frac{\ln(t)+1}{4}$ and $\sum_{\tau=1}^{t}\eta^{(\tau)} = \frac{1}{2}\sum_{\tau=1}^{t}\sqrt{\frac{1}{\tau}} \geq \sqrt{t}/2$ we find that, with probability at least $1-\frac{\delta}{2t^2}$, we have for (5.3):

$$\begin{aligned}\forall i \in [n], j \in [m] :&\left(Ax^{(t)}/\|x^{(t)}\|_1\right)_j - \left(A^T y^{(t)}/\|y^{(t)}\|_1\right)_i \\ &\leq \frac{\ln\left(2t^2 nm/\delta\right) + (\ln(t)+1)}{\sqrt{t}/2} \\ &\leq \frac{2}{\sqrt{t}} \cdot (3\ln(t) + \ln(nm) + \ln(1/\delta) + 2).\end{aligned}$$

Taking the union bound over the error probabilities we have that the total error probability over all iterations is less than $\sum_{t\in\mathbb{N}}\frac{\delta}{2t^2} = \frac{\delta}{2}\cdot\frac{\pi^2}{6} \leq \delta$.

Now we finish the analysis of the fixed-error version of the algorithm choosing $\eta^{(t)} := \varepsilon/4$ and $\delta^{(t)} := \delta$. In this setting we will not use the union bound since we only require the last iteration to be correct. With probability at least $1-\delta$ we have for (5.3)

$$\begin{aligned}\forall i \in [n], j \in [m] :&\left(Ax^{(t)}/\|x^{(t)}\|_1\right)_j - \left(A^T y^{(t)}/\|y^{(t)}\|_1\right)_i \\ &\leq \frac{\ln\left(\frac{nm}{\delta}\right) + 3t\varepsilon^2/16}{t\varepsilon/4} \\ &= \frac{3}{4}\varepsilon + \frac{4\ln\left(\frac{nm}{\delta}\right)}{t\varepsilon}.\end{aligned}$$

For $t \geq \frac{16\ln\left(\frac{nm}{\delta}\right)}{\varepsilon^2}$ this can be further upper bounded by ε. □

Once we get the ε-optimal solutions we can estimate the value of the game by simply playing the game with the corresponding randomized strategies. Since in each run the game has bounded value, by Chernoff's bound we get the following:

Claim 5.2 *Given a pair of strategies x, y, let us take $k = \mathcal{O}\left(1/\varepsilon^2\right)$ independent samples $i_1, \ldots, i_k$ from x and similarly $j_1, \ldots, j_k$ from y. Then, with bounded error probability, the average $\sum_{\ell=1}^{k} A_{j_\ell, i_\ell}/k$ is an ε-approximate estimate of the value of the game corresponding to these strategies.*

This clearly shows that obtaining the approximate solutions via Algorithm 5.1 dominates the complexity of approximately computing the corresponding value.

If access to A is given by an oracle for the entries of A, then the following lemma gives a bound on the cost of the algorithm for finding the optimal strategies.

Lemma 5.3 *Algorithm 5.1 can be implemented to find an ε-optimal pair of solutions on a classical randomized computer with probability at least $1-\delta$ in $\frac{16}{\varepsilon^2}\ln\left(\frac{nm}{\delta}\right)$ iterations, using $\widetilde{\mathcal{O}}_{\frac{1}{\varepsilon}}(n+m)$ time per iteration, and $n+m$ queries to the entries of A per iteration.*

Proof. The iteration bound follows from Lemma 5.1. In the rest of the proof we drop the (t) superscript for ease of notation. Since in each iteration only one entry of x changes, the change in u can be computed using m queries. It then requires $\widetilde{\mathcal{O}}_{\frac{1}{\varepsilon}}(m)$ work to compute P and p. In the same amount of time the cumulative distribution corresponding to p can be calculated. By generating a random number between 0 and 1 and performing binary search on the cumulative distribution we can sample a from p. A similar approach works for y, v, Q, q and b. □

When A is given by a *sparse oracle* which also allows querying for any j the location of the j-th non-zero entry in each row and column, then a further speedup is possible.

Lemma 5.4 *Let s be the maximum number of non-zero entries in a column of A, and d the maximum number of non-zero entries in a row of A. Algorithm 5.1 can be used to find an ε-optimal pair of solutions on a classical randomized computer with probability at least $1-\delta$ in $\frac{16}{\varepsilon^2}\ln\left(\frac{nm}{\delta}\right)$ iterations, $\widetilde{\mathcal{O}}_{\frac{nm}{\varepsilon}}(s+d)$ time per iteration, and $s+d$ queries to a sparse oracle for A per iteration.*

Proof. The iteration bound follows from Lemma 5.1. In the rest of the proof we drop the (t) superscript for ease of notation. Now, since there is a sparse oracle for the columns of A, the change in u can be computed using s queries and $\widetilde{\mathcal{O}}_{\frac{nm}{\varepsilon}}(s)$ time. Hence, also the multiplicative change in P can be calculated in $\widetilde{\mathcal{O}}_{\frac{nm}{\varepsilon}}(s)$ time.

Now, instead of keeping P as a list, we keep it in the tree data structure from Lemma 4.27. Updating one leaf can be done in $\widetilde{\mathcal{O}}_{\frac{nm}{\varepsilon}}(1)$ time, so P can be updated in $\widetilde{\mathcal{O}}_{\frac{nm}{\varepsilon}}(s)$ time. Given a tree structure for the vector P, sampling from $P/\|P\|_1$ can easily be implemented. A similar approach works for y, v, Q, q and b and the lemma statement follows. □

5.3 | Quantum implementation of the algorithm

In the quantum case we aim at a sublinear-time algorithm in m and n. This means that we cannot even read through a single column or row of A. We define two types of quantum query access for the matrix A. In the *dense input model* we assume access to an oracle that returns a binary representation of the queried matrix entry, similar to an oracle of the form (4.7) in the SDP setting. In the *sparse input model* we additionally assume access to an oracle that returns positions of the non-zero entries in a row or column, similar to an oracle of the form (4.6) in the SDP setting.

Observe that Algorithm 5.1 is very similar to the SDP-solving meta-algorithms of Chapter 4. In each iteration we take a linear combination of inputs (now the rows or columns of A) and consider the distribution proportional to the exponent of this linear combination. There are two differences that make the quantum algorithm for zero-sum games simpler than those for SDP-solving. The first difference is that only a single sample from the exponential distribution is needed, while in the SDP frameworks we required many more samples in order to estimate the trace values. The second difference is that we are dealing with vectors instead of matrices. This leads us to the following definition.

Definition 5.5 : Classical Gibbs distribution *For a vector $v \in \mathbb{R}^n$ let e^v denote the vector whose i-th coordinate is e^{v_i}. Then $G(v) := \frac{e^v}{\|e^v\|_1}$ denotes the* Gibbs distribution *corresponding to v.*

The algorithm always maintains $x^{(t)}$ and $y^{(t)}$ that have at most t non-zero elements. We store $x^{(t)}$ and $y^{(t)}$ in the tree data structure from Lemma 4.27, similar to how we stored $y^{(t)}$ in the SDP algorithms. This enables us to query elements, sample from the distributions that are proportional to the vectors, and create state-preparation pairs for $x^{(t)}$ and $y^{(t)}$. Implementing one iteration of Algorithm 5.1 essentially boils down to efficient Gibbs-sampling, i.e., implementing a single sampling from the distribution $e^{Ax}/\|e^{Ax}\|_1$.

5.3.1 | Dense matrices

In this section we will assume that all the entries in A are non-negative. This is without loss of generality since we can consider $A' = \frac{1}{2}(J + A)$, where J is the all-ones matrix. An $\frac{\varepsilon}{2}$-approximate solution for A' is an ε-approximate solution for A.

Before we state the quantum algorithm, let us describe a classical algorithm that mirrors its behavior. We will focus on the t-th iteration of Algorithm 5.1. We will write x instead of $x^{(t)}$ for simplicity. We assume that x is t-sparse and is stored in the tree data structure from Lemma 4.27, our goal is to sample from the Gibbs distribution $G(Ax)$. Since x is t-sparse we can compute a single element of $u = Ax$ using $\widetilde{\mathcal{O}}_n(t)$ steps. Since we have a procedure to calculate u_j, we can find the maximal element u_j using $\mathcal{O}(m)$ calls to this procedure. Call this maximum $u_{\max}$. We will sample from $G(Ax - u_{\max}\mathbf{1}) = G(Ax)$ since this allows us to use that $e^{u_j - u_{\max}} \leq 1$ for all j. To do this sampling we will use rejection sampling: sample $j \in [m]$ uniformly at random, then with probability $e^{u_j - u_{\max}}$ output j and with probability $1 - e^{u_j - u_{\max}}$ output $\perp$. If we would post-select on the outcome not being $\perp$, then we would have sampled from $G(Ax - u_{\max}\mathbf{1}) = G(Ax)$.

Hence we repeat this procedure an expected $\mathcal{O}\left(\frac{me^{u_{\max}}}{\sum_{j=1}^m e^{u_j}}\right) \leq \mathcal{O}(m)$ times until we get an output other than $\perp$.

All the steps described above have quadratically faster quantum counterparts. Estimating a single value u_j can be done via amplitude estimation, finding the maximum can be done by the Generalized Minimum-Finding Lemma (Lemma 4.9), and rejection sampling can be done in $\mathcal{O}(\sqrt{m})$ steps via amplitude amplification.

All these methods are already present in Chapter 4 by only considering diagonal matrices. However, we will prove them again below since the proofs are simpler when considering vectors. For this we will use the following lemma to implement block-encodings of diagonal matrices:

Lemma 5.6 *Let $v \in \mathbb{R}^n_{\geq 0}$ with $\|v\|_\infty \leq 1$ and let $a \in \mathbb{N}$. Let U be a unitary that on input $|0^{a+1}\rangle|j\rangle$ prepares a quantum state $\left(\sqrt{v_j}|0\rangle|\psi_j\rangle + \sqrt{1-v_j}|1\rangle|\phi_j\rangle\right)|j\rangle$ for some states $|\psi_j\rangle$ and $|\phi_j\rangle$. Then a single query to both U and $U^\dagger$, and a single SWAP gate, suffice to implement a $(1, a+2, 0)$-block-encoding of* $\mathrm{diag}(v)$.

Proof. Let SWAP_{12} be the gate that swaps the first two qubits. Now observe that for

$$V := (I \otimes U^\dagger)(\mathrm{SWAP}_{12} \otimes I)(I \otimes U)$$

we have

$$\left(\langle 0^{a+2}| \otimes I\right) V \left(|0^{a+2}\rangle \otimes I\right) = \mathrm{diag}(v),$$

with a similar argument to Lemma 4.24. Hence V is a $(1, a+2, 0)$-block-encoding of the matrix $\mathrm{diag}(v)$. □

We are now ready to prove the first of our two quantum results.

Lemma 5.7 *Suppose that $x \in \mathbb{R}^n$ is stored in QCRAM using the data structure from Lemma 4.27, such that $\|x\|_1 \leq \beta$ for some known $\beta \geq 1$. If we have quantum query access to a dense oracle for the matrix elements of $A \in [0,1]^{n\times m}$, then we can sample from a distribution that is μ-close in total variation distance to $G(Ax)$, using $\widetilde{\mathcal{O}}_{\frac{n}{\mu}}\left(\beta\sqrt{m}\right)$ quantum queries to A and elementary operations.*

Proof. Using the tree data structure we can implement a unitary in $\widetilde{\mathcal{O}}_{\frac{n\beta}{\xi}}(1)$ elementary operations that maps

$$|0\rangle|0\rangle \mapsto |0\rangle \sum_{i\in[n]} \sqrt{x_i/\beta}|i\rangle + |1\rangle|\phi\rangle,$$

up to an ℓ_2-norm error $\mathcal{O}(\xi)$ in the result. Here $|\phi\rangle$ is some subnormalized quantum state on a (possibly multi-qubit) register. Similarly, with a single additional query to A, and $\widetilde{\mathcal{O}}_{\frac{m}{\xi}}(1)$ additional elementary operations, we can implement a unitary U that on input $|0\rangle|0\rangle|j\rangle$ prepares a state that is $\mathcal{O}(\xi)$-close in ℓ_2-norm to

$$\left(|0\rangle \sum_{i\in[n]} \sqrt{A_{ji}x_i/\beta}|i\rangle + |1\rangle|\phi'\rangle\right)|j\rangle.$$

The probability of getting outcome 0 when measuring the first register in the computational basis is $\mathcal{O}(\xi)$-close to u_j/β, hence we can $\frac{1}{4}$-approximate (if $\xi = \mathcal{O}(1/\beta)$) a single u_j with bounded error probability using amplitude estimation, using $\widetilde{\mathcal{O}}_{\frac{nm}{\xi}}(\beta)$ queries and elementary operations.

Let $u_{\max} := \max_{j\in[m]} u_j$. We can also find a constant additive approximation $\tilde{u}_{\max} \in [u_{\max}, u_{\max}+1]$ using $\widetilde{\mathcal{O}}_{\frac{n}{\xi}}(\beta\sqrt{m})$ queries and elementary operations with bounded error probability using Generalized Quantum Minimum-Finding (Lemma 4.9). We can boost the success probability to $1-\frac{\mu}{3}$ with $\mathcal{O}(\log(1/\mu))$ repetitions, hence we will for the rest of the proof assume this process was successful.

Using Lemma 5.6 with the unitary U we can implement a $\left(\beta, \widetilde{\mathcal{O}}_{\frac{mn\beta}{\xi}}(1), \mathcal{O}(\xi)\right)$-block-encoding of $\mathrm{diag}(Ax)$. Using the Linear combination of block-encodings lemma (Lemma 2.9) we can then also implement U', a $\left(2\beta, \widetilde{\mathcal{O}}_{\frac{mn\beta}{\xi}}(1), \mathcal{O}(\xi)\right)$-block-encoding of $M := \mathrm{diag}(Ax - \tilde{u}_{\max}\mathbf{1})$.

Now we are ready to implement the rejection sampling. We first prepare the uniform superposition $\frac{1}{\sqrt{m}}\sum_{j\in[m]}|j\rangle$ using $\widetilde{\mathcal{O}}_m(1)$ elementary operations. Then ideally we would like to apply the map

$$\begin{aligned}
|0\rangle|j\rangle \mapsto \sqrt{e^{u_j-\tilde{u}_{\max}}}|0\rangle|j\rangle + |1\rangle|\perp''\rangle \\
= |0\rangle e^{\beta\frac{\mathrm{diag}(Ax-\tilde{u}_{\max}\mathbf{1})}{2\beta}}|j\rangle + |1\rangle|\perp''\rangle \\
= |0\rangle e^{M/2}|j\rangle + |1\rangle|\perp''\rangle
\end{aligned}$$

to this uniform superposition.

We implement a good approximation of the above by replacing the function $e^{\beta z}$ with an approximating polynomial $P(z)$ and using Lemma 2.11. For this we use the approximation polynomial $P(z)$ from Lemma 4.14. We can then prepare a state that is $\widetilde{\mathcal{O}}(\sqrt{\beta\xi})$-close in ℓ_2-norm to

$$\frac{1}{\sqrt{m}}\left(\sum_{j\in[m]}|0\rangle P\left(\frac{\mathrm{diag}(Ax-\tilde{u}_{\max}\mathbf{1})}{2\beta}\right)|j\rangle\right) + |1\rangle|\phi'''\rangle,$$

for some arbitrary subnormalized state $|\phi''\rangle$. This will use the block-encoding for M a total of $\widetilde{\mathcal{O}}(\deg(P)) = \widetilde{\mathcal{O}}(\beta\log(1/\xi))$ times, as shown by Lemma 2.11. Since $P(z)$ is an $\mathcal{O}(\xi)$-approximation of $e^{\beta z}/4$, this last state is $\mathcal{O}(\sqrt{\beta\xi})$-close in ℓ_2-norm to

$$\frac{1}{\sqrt{m}}\left(\sum_{j\in[m]}|0\rangle e^{\beta\frac{\mathrm{diag}(Ax-\tilde{u}_{\max}\mathbf{1})}{2\beta}}/4|j\rangle\right) + |1\rangle|\phi'''\rangle.$$

We now consider the probability of getting outcome 0 when measuring the first register in the computational basis. Since there is at least one j in the sum for which the amplitude is $\frac{e^{-\frac{1}{8}}}{4\sqrt{m}}$, namely the j corresponding to $u_{\max}$, we know that this probability is at least $\Omega(1/m)$ (if $\beta\xi$ is small enough).

Finally, we obtain a sample with probability at least $1-\frac{\mu}{9}$ with $\mathcal{O}\left(\sqrt{m}\log(1/\mu)\right)$ applications of the full procedure (and its inverse) using amplitude amplification. If we pick $\xi = \widetilde{\mathcal{O}}\left(\frac{\mu^2}{\beta m^2}\right)$ then we sample from a distribution that is μ-close to $G(Ax)$ in total variation distance. In total we used $\widetilde{\mathcal{O}}_{\frac{n}{\mu}}\left(\sqrt{m}\beta\right)$ queries and elementary operations. □

Theorem 5.8 *Algorithm 5.1 can be used to find an ε-optimal pair of solutions with probability at least $1-\delta$ in $\frac{16}{\varepsilon^2}\ln\left(\frac{nm}{\delta}\right)$ iterations. On a quantum computer with QCRAM the t-th iteration can be implemented in $\widetilde{\mathcal{O}}_{\frac{1}{\delta}}\left((1+t\varepsilon)\sqrt{n+m}\right)$ quantum queries to the entries of A and the same number of elementary operations, leading to a total of $\widetilde{\mathcal{O}}_{\frac{1}{\delta}}\left(\sqrt{n+m}/\varepsilon^3\right)$ queries and elementary operations.*

Proof. We use the fixed-error version of Algorithm 5.1. The theorem follows from Lemma 5.1 and Lemma 5.7, by setting the error μ in total variation distance to $\mathcal{O}\left(\varepsilon^2\delta/\ln(nm/\delta)\right)$ in the latter. □

5.3.2 | Sparse matrices

For sparse matrices we can no longer assume that all entries in A are non-negative, since adding the all-ones matrix to A would result in a dense matrix. To sample from the Gibbs distribution for sparse matrices we will use the following polynomial approximation lemma.

Lemma 5.9 : [GSLW19, Cor. 67] *Let $\xi \in (0,1/6]$ and $\beta \geq 1$. There exists a polynomial $R(x)$ such that*

- *For all $x \in [\frac{1}{2\beta},1]$ we have $\left|R(x)-\frac{1}{\sqrt{4\beta x}}\right| \leq \xi$.*
- *For all $x \in [-1,1]$ we have $|R(x)| = |R(-x)| \leq 1$.*
- $\deg(R) = \widetilde{\mathcal{O}}_{\frac{1}{\xi}}(\beta)$.

We state our next result with s denoting an upper bound on the sparsity of both the rows and the columns of A. Note that if separate bounds are known for each, then s is simply the maximum of the two bounds.

Lemma 5.10 *Suppose $x \in \mathbb{R}^n_{\geq 0}$ is stored in the data structure from Lemma 4.27, and $\|x\|_1 \leq \beta$ for some known $\beta \geq 1$. If we have quantum query access to a sparse oracle for $A \in [-1,1]^{m\times n}$, and if A has s-sparse rows and columns, then we can sample from a distribution that is μ-close to $G(Ax)$ in total variation distance, using $\widetilde{\mathcal{O}}_{\frac{mn}{\mu}}\left(\beta^{\frac{3}{2}}\sqrt{s}\right)$ quantum queries and elementary operations.*

Proof. We can assume without loss of generality that $\beta s \leq m/4$, otherwise the statement follows from Lemma 5.7. Let us define $w \in \mathbb{R}^m$ by

$$w_j := \sum_{i\in[n]} |A_{ji}|x_i \geq u_j.$$

While u is no longer guaranteed to be non-negative, w_j is.

In Lemma 5.7 we used a quantum version of rejections sampling, starting from the uniform distribution and “accepting” an outcome with a probability proportional to the exponent of that outcome. Since the uniform distribution might be far away in total variation distance from the Gibbs distribution, this process required $\widetilde{\mathcal{O}}(\sqrt{m})$ amplitude amplification steps. The main idea is to again use rejection sampling: for the parts of the Gibbs distribution that are close to uniform we again start from the uniform distribution, but for the parts that are far from uniform we start from a distribution proportional to w_j.

As $\sum_{j\in[m]} w_j \leq \beta s \leq m/2$ we know that $w_j \leq 1$ for at least half of the js. For such j we have $|u_j| \leq 1$ and hence the distribution $G(u)$ is quite uniform on these positions. On the other hand, we can sample from a distribution proportional to w_j by first sampling an i with probability x_i/β and then sampling a j with probability $\frac{|A_{ji}|}{s}$. For the js where $w_j \geq 1/2$ this distribution is not too far from the Gibbs distribution and we will only use $\widetilde{\mathcal{O}}(\sqrt{s\beta})$ amplitude amplification steps. Since the two regimes cover every $j \in [m]$ we can Gibbs sample a j by combining the two sampling procedures, in a similar fashion to [BKLLSW19; AG19a].

The proof is structured as follows. We start by showing how to implement a block-encoding of u, and how to approximate the entries of u. We then show how to approximate $u_{\max}$ to constant additive error. After this we give a method for distinguishing the two regimes. We then show how to Gibbs sample for both the small and large values, and finish by combining these samples.

A block-encoding and approximation for u In order to implement a block-encoding of $\mathrm{diag}(u)$ we modify the proof of Lemma 5.6. Let U_R be a unitary that, starting from $|0\rangle|0\rangle|0\rangle|j\rangle$, prepares a state that is $\mathcal{O}(\xi)$-close in ℓ_2-norm to

$$|0\rangle\left(|0\rangle\sum_{i\in[n]}\mathrm{sign}(A_{ji})\sqrt{|A_{ji}|x_i/\beta}|i\rangle+|1\rangle|\phi'\rangle\right)|j\rangle.$$

We can implement such a unitary with a similar argument to that of Lemma 5.7. Furthermore, let U_L be a unitary that, starting from $|0\rangle|0\rangle|0\rangle|j\rangle$, prepares a state that is $\mathcal{O}(\xi)$-close in ℓ_2-norm to

$$U_L : |0\rangle|0\rangle|0\rangle|j\rangle \mapsto \left(|0\rangle\sum_{i\in[n]}\sqrt{|A_{ji}|x_i/\beta}|i\rangle+|1\rangle|\phi''\rangle\right)|j\rangle.$$

Then

$$V := U_L^\dagger(\mathrm{SWAP}_{12}\otimes I)U_R$$

is a $(\beta, \widetilde{\mathcal{O}}_{\frac{mn\beta}{\xi}}(1), \mathcal{O}(\xi))$-block-encoding of $\mathrm{diag}(Ax) = \mathrm{diag}(u)$.

Using Hamiltonian simulation and phase estimation we can approximate u_j with precision $\frac{1}{4}$ using $\widetilde{\mathcal{O}}_{\frac{mn\beta}{\varepsilon}}(\beta)$ queries to A and elementary operations in total.

Finding the maximal element We start by assuming that $u_{\max}$ is greater than 1. As in Lemma 5.7 we use the tree data structure for x to prepare a state that is $\mathcal{O}(\xi)$-close in ℓ_2-norm to

$$|0\rangle\left(\sum_{i\in[n]}\sqrt{x_i/\beta}|i\rangle\right)+|1\rangle|\phi\rangle,$$

for some arbitrary subnormalized state $|\phi\rangle$, using $\widetilde{\mathcal{O}}_{\frac{nm}{\xi}}(1)$ elementary operations. With a single additional query, we can prepare a state that is $\mathcal{O}(\xi)$-close in ℓ_2-norm to

$$|\psi\rangle := |0\rangle\left(\sum_{i\in[n]}\sum_{j:\ A_{ji}\neq 0}\sqrt{|A_{ji}|x_i/(\beta s)}|i\rangle|j\rangle\right)+|1\rangle|\phi'\rangle,$$

for some arbitrary subnormalized $|\phi'\rangle$. Observe that if $u_j > 1$, then measuring the first and last registers of $|\psi\rangle$ would, with probability at least $w_j/(2\beta s) \geq u_j/(2\beta s) > 1/(2\beta s)$, result in the outcomes 0 and j. We now add a new register, and for the $|0\rangle|i\rangle|j\rangle$ part of $|\psi\rangle$ we approximate u_j up to additive error $1/4$ using a total of $\widetilde{\mathcal{O}}_{\frac{nm}{\xi}}(\beta)$ queries and elementary operations.

We can find a constant additive approximation $\tilde{u}_{\max} \in [u_{\max}, u_{\max}+1]$ of the maximum value by using the Generalized Quantum Minimum-Finding algorithm (Lemma 4.9). This will use $\widetilde{\mathcal{O}}_{\frac{nm}{\xi}}\left(\sqrt{s}\beta^{3/2}\right)$ queries and elementary operations in total to succeed with $2/3$ probability, and (as in Lemma 5.7) we can boost the success probability to $1-\mu/3$ with $\mathcal{O}\left(\log(1/\mu)\right)$ repetitions.

Now if $u_{\max}$ is less than 1, then the procedure above will result in a $\tilde{u}_{\max}$ below 2 (that might not be close to the actual maximum). However, since $w_j \leq 1$ for at least half of the j, we know that if we get such a result, then $u_j \in [-1,2]$ for at least half of the j (and no u_j is larger than 2). This implies that the Gibbs distribution is very close to uniform, and that we do not need to make the distinction between small and large values of w_j. Instead we can use the argument from Lemma 5.7, with the difference that we only need a constant number of amplitude amplification steps. For the rest of the proof we therefore assume that $u_{\max} > 1$.

Distinguishing small and large values To distinguish the j with a large w_j from those with a small w_j we would ideally implement a unitary that sets the second register of $|j\rangle|0\rangle$ to 1 if and only if $w_j \geq 3/4$. This would be a block-encoding of the projection on all j such that $w_j < 3/4$. We cannot implement this procedure exactly, but we can set the flag correctly with bounded error probability for all j where $w_j \notin [1/2, 1]$.

Consider the unitary U_L from before. Using U_L we can approximate w_j up to additive error $1/8$ with bounded error probability using $\widetilde{\mathcal{O}}_{\frac{nm}{\xi}}(\beta)$ queries and elementary operations. We can boost the success probability exponentially in the number of repetitions, hence we will for the rest of the proof assume that this procedure never fails. If we now set a flag depending on our approximation of w_j then the flag will be correctly set for all j where $w_j \notin [1/2,1]$. We denote the unitary that on input $|j\rangle|0\rangle$ sets the second register in this way by Q.

It is important to note that for $w_j \in [1/2, 1]$ this unitary Q can set the flag to an arbitrary superposition of $|0\rangle$ and $|1\rangle$. However, for a specific j this will always be the same superposition. Hence, if we use this flag to either include j when Gibbs sampling the small values, or to include j when Gibbs sampling the large values, then in the end j will still be sampled with the correct probability.

Gibbs-sampling for the small values To Gibbs-sample for the small w_j, consider the following procedure. First (using $\widetilde{\mathcal{O}}_m(1)$ elementary operations) prepare the state

$$\frac{1}{\sqrt{m}}\sum_{j\in[m]}|j\rangle|0\rangle|0\rangle.$$

Then apply Q to the first and second register to set a flag. For the part of the superposition where this flag is 0 we have that $w_j \leq 1$ for the corresponding j, if this flag is 1, then $w_j \geq 1/2$.

For now we only consider the part of the state corresponding to the first case, which implies that $u_j \leq w_j \leq 1$. We apply the map $\mathrm{diag}\left(e^{\frac{u-1}{2}}/4\right)$ to the first register, indicating "success" with the third register being set to 0. This can be done with $\widetilde{\mathcal{O}}_{\frac{mn}{\xi}}(\beta)$ queries and elementary gates in total, similarly to Lemma 5.7.

The probability of getting outcome 0 twice when measuring both flags, and getting outcome j for the first register now is (for a j where $w_j \leq 1/2$)

$$\frac{1}{m}\cdot\frac{e^{u_j}}{32e} \geq \frac{e^{-|u_j|}}{32em} \geq \frac{e^{-w_j}}{32em} \geq \frac{1}{32e^2 m}.$$

There are at least $m/4$ such j, so the probability of getting outcome 0 for both flags (indicating success) is constant.

Gibbs-sampling for the large values Now, to Gibbs-sample for the large w_js we consider a similar procedure starting from the state $|\psi\rangle$ instead of the maximally entangled state. We rewrite this state as

$$|\psi\rangle = |0\rangle\left(\sum_{j\in[n]}\sqrt{\frac{w_j}{\beta s}}|\psi_j\rangle|j\rangle\right) + |1\rangle|\phi''\rangle$$

for some arbitrary subnormalized state $|\phi''\rangle$ and some (normalized) states $|\psi_j\rangle$. We will consider the part of $|\psi\rangle$ where the first qubit is in the $|0\rangle$ state as "success". As before we apply Q, now to the third register (and some ancilla qubits), but now we care about the part of the state where the resulting flag is 1.

Now we will apply $\frac{e^{(u_j-\tilde{u}_{\max})/2}}{8\sqrt{w_j}}$ in two steps. First we apply the map $e^{(u_j-\tilde{u}_{\max})/2}/4$ to the third register as before. Next, we can implement a $\left(1, \widetilde{\mathcal{O}}_{\frac{\beta nm}{\xi}}(1), \widetilde{\mathcal{O}}(\xi)\right)$-block-encoding of $\mathrm{diag}(w/\beta)$ using Lemma 5.6 with the unitary U_L. Using the polynomial R

from Lemma 5.9, and with Lemma 2.11, we can apply a $\left(1,\widetilde{\mathcal{O}}_{\frac{\beta nm}{\xi}}(1),\widetilde{\mathcal{O}}\left(\sqrt{\beta\xi}\right)\right)$-block-encoding of $\mathrm{diag}\left(1/\sqrt{4w}\right)$ to the third register.

In total we get a state that is $\widetilde{\mathcal{O}}\left(\sqrt{\beta\xi}\right)$-close in ℓ_2-norm to

$$|0\rangle\left(\sum_{j\in[n]}\frac{e^{(u_j-\tilde{u}_{\max})/2}}{8\sqrt{\beta s}}|\psi_j\rangle\left(Q|j\rangle|0\rangle\right)\right)+|1\rangle|\phi'''\rangle$$

using $\widetilde{\mathcal{O}}_{\frac{nm}{\xi}}(\beta)$ queries and elementary operations. Now for the j corresponding to $u_{\max}$ we know that $e^{(u_j-\tilde{u}_{\max})/2}=\widetilde{\Omega}_{\frac{\xi}{\beta s}}(1)$, and that the last register is in the state $|1\rangle$. Therefore if we would measure the first and last registers, then the probability of getting outcomes 0 and 1 respectively (indicating success) is at least $\widetilde{\Omega}_{\xi}(1/\beta s)$.

Combining the samples Finally, notice that the two resulting partial Gibbs states are subnormalized in different (but known) ways. Let us define $N:=16em+64\beta se^{\tilde{u}_{\max}}$. We sample a j from the full distribution in the following way: with probability $\frac{16em}{N}$ sample j using the first procedure, and with probability $\frac{64\beta se^{\tilde{u}_{\max}}}{N}$ sample j using the second procedure. Since both procedures have total success probability $\widetilde{\Omega}_{\xi}\left(\frac{1}{\beta s}\right)$ the success probability of the final algorithm is also $\widetilde{\Omega}_{\xi}\left(\frac{1}{\beta s}\right)$. Therefore we can sample a j with $\mathcal{O}\left(\sqrt{\beta s}\right)$ rounds of amplitude amplification with bounded error probability, and we can boost the success probability to $1-\mu/9$ by $\mathcal{O}\left(\log(1/\mu)\right)$ repetitions. If we set $\xi=\mathcal{O}\left(\frac{\mu}{m\beta^2 s}\right)=\mathcal{O}\left(\frac{\mu}{\beta m^2}\right)$, then the distribution will be μ-close to the true Gibbs distribution. The complexity of each round of amplitude amplification is $\widetilde{\mathcal{O}}_{\frac{mn}{\mu}}(\beta)$, which leads to the final complexity statement. □

Theorem 5.11 *Algorithm 5.1 can be used to find an ε-optimal pair of solutions with probability at least $1-\delta$ in $\frac{16}{\varepsilon^2}\ln\left(\frac{nm}{\delta}\right)$ iterations. On a quantum computer with QCRAM the t-th iteration can be implemented using $\widetilde{\mathcal{O}}_{\frac{nm}{\delta}}\left((1+t\varepsilon)^{\frac{3}{2}}\sqrt{s}\right)$ elementary operations and quantum queries to a sparse oracle of A, where A has s-sparse rows and columns, giving a total of $\widetilde{\mathcal{O}}_{\frac{mn}{\delta}}\left(\sqrt{s}/\varepsilon^{3.5}\right)$ queries and elementary operations.*

Proof. This follows from Lemma 5.1 and Lemma 5.7, by setting the error in total variation distance μ to $\mathcal{O}\left(\varepsilon^2\delta/\ln(nm/\delta)\right)$ in the latter. □

5.4 | Reduction of LP-solving to zero-sum games

In this section we will reduce general LPs to zero-sum games to obtain a faster quantum algorithm for LP-solving. A similar argument can be found in for example [CDST19]. We consider an LP in the standard form as introduced in Section 2.4.2:

$$\begin{aligned} \max_{x\in\mathbb{R}^n} \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{5.4}$$

with the dual LP

$$\begin{aligned} \min_{y\in\mathbb{R}^m} \quad & b^T y \\ \text{s.t.} \quad & A^T y \geq c \\ & y \geq 0. \end{aligned} \tag{5.5}$$

We assume without loss of generality that all entries of A and c are in $[-1,1]$. We assume that both LPs are feasible. It follows that strong duality holds and the two optimal values coincide. We will write OPT for this optimal value. Furthermore, we assume bounds R and r are known that are similar to the bounds in the SDP setting: adding the constraint $\sum_{i=1}^n x_i \leq R$ to the primal will not change the optimal value of the primal and adding the constraint $\sum_{j=1}^m y_j \leq r$ to the dual will not change the optimal value of the dual. We can assume without loss of generality that $|b_j| \leq R$ for all $j \in [m]$, since if $b_j < -R$, then the LP would be infeasible, and if $b_j > R$, then we could remove the constraint without changing the value of OPT.[3] We will assume that $R, r \geq 1$. Finally, let s denote a bound on the row and column sparsity of A, as well as on the sparsity of b and c. Our reduction will consist of the following steps:

1. Reduce an optimization problem to feasibility using binary search.
2. Scale the problem such that the solutions will be in the simplex.
3. Reduce to a problem where all right-hand sides of the inequality constrains are 0.
4. Reduce to a zero-sum game.

Reduction to feasibility Note that $-R \leq \text{OPT} \leq R$, because $\|c\|_\infty \leq 1$ and there is an optimal solution with $\|x\|_1 \leq R$. Hence, if we can answer questions of the type "is OPT $\geq \alpha - \varepsilon$ or OPT $\leq \alpha$" then we can use $\log(R/\varepsilon)$ iterations of binary search to determine OPT up to additive error ε. To answer questions of this form we add $c^T x \geq \alpha$ as a constraint and ask whether there is a feasible x. That is, we want to know whether

[3] Note that for a sublinear algorithm we do not actually have time to remove a constraint. In fact, in an oracle model it is not even clear what this would mean. However, we can implement a new oracle for the values A_{ji}, with one additional query to b, that returns 0 if $b_j > R$, effectively removing the constraint.

there is no x satisfying

$$\begin{aligned} -c^T x &\leq -\alpha \\ Ax &\leq b \\ \sum_{i=1}^{n} x_i &\leq R \\ x &\geq 0, \end{aligned}$$

indicating that OPT ≤ α, or that[4] (for some fixed $\varepsilon^{(1)}$) there is an x such that

$$\begin{aligned} -c^T x &\leq -\alpha + \varepsilon^{(1)} \\ Ax &\leq b + \mathbf{1}\varepsilon^{(1)} \\ \sum_{i=1}^{n} x_i &\leq R \\ x &\geq 0, \end{aligned}$$

indicate that OPT $\geq \alpha - \varepsilon$. Here we need to be careful with our choice of $\varepsilon^{(1)}$ since relaxing all the constraints by $\varepsilon^{(1)}$ might change the value of OPT by as much as $\varepsilon^{(1)}(r+1)$, as argued in Section 4.1.1. Hence we pick $\varepsilon^{(1)} = \Theta(\varepsilon/(r+1))$.

Scale to the simplex Note that by dividing all the right-hand sides by R we simply scale down x such that $\sum_i x_i \leq 1$. This, however, does imply that we want a lower additive error: $\varepsilon^{(2)} = \Theta(\varepsilon/(R(r+1)))$. Let us define $x^{(2)} := (x, z)^T$ where $z \in \mathbb{R}$ is a new variable. Then we obtain the new feasibility problem

$$\begin{aligned} \exists ? x^{(2)} \in \Delta^{n+1} \quad & s.t. \\ & A^{(2)} x^{(2)} \leq b^{(2)}, \end{aligned}$$

where

$$A^{(2)} = \begin{pmatrix} -c^T & 0 \\ A & 0 \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} -\alpha/R \\ b/R \end{pmatrix}.$$

Right-hand sides zero For the final reduction to a zero-sum game, we want that all right-hand sides are equal to zero. To achieve this we use that $\sum_i x_i^{(2)} = 1$. We introduce an extra variable h and add the constraint $\sum_i x_i^{(2)} = h$. If we now constrain the new vector $x^{(3)} = (x^{(2)}, h)^T$ to be in the simplex (instead of $x^{(2)}$ being in the simplex) then we find that $\sum_i x_i^{(2)} = h = 1/2$. Hence by scaling all the right-hand sides by a factor $1/2$ we will not have changed whether the LP is feasible. But now we have a variable h that is fixed to a constant, so we can shift the inequalities by setting the right-hand side to zero

[4] These cases are potentially overlapping. In the intersection we are happy with either conclusion.

while appropriately changing the coefficient of h on the left-hand side. In particular we get the new feasibility problem

$$\begin{aligned} \exists ? x^{(3)} \in \Delta^{n+2} \quad & s.t. \\ & A^{(3)} x^{(3)} \leq 0, \end{aligned}$$

where

$$A^{(3)} = \begin{pmatrix} \mathbf{1}^T & -1 \\ -\mathbf{1}^T & 1 \\ A^{(2)} & -b^{(2)} \end{pmatrix}, \quad x^{(3)} = \begin{pmatrix} x^{(2)} \\ h \end{pmatrix}.$$

It suffices to bring down the additive error by a factor of two in order to solve the previous feasibility problem using this new problem.

A zero-sum game To construct a zero-sum game as in (5.1) we simply observe that

$$\begin{aligned} \min_{x^{(3)} \in \Delta^{n+2}, \lambda \in \mathbb{R}} \quad & \lambda \\ \text{s.t.} \quad & A^{(3)} x^{(3)} \leq \lambda \mathbf{1} \end{aligned}$$

has a value less than $\varepsilon^{(3)}$ if and only if there is a point that is $\varepsilon^{(3)}$-feasible for the last LP.

The final game matrix now is

$$A^{(3)} = \begin{pmatrix} \mathbf{1}^T & 1 & -1 \\ -\mathbf{1}^T & 1 & 1 \\ -c^T & 0 & \alpha/R \\ A & 0 & -b/R \end{pmatrix}. \tag{5.6}$$

Now let us prove that the sketch above indeed gives the desired result.

Lemma 5.12 *Finding the optimal value λ^* of the game* (5.6) *up to additive error $\varepsilon^{(3)} = \varepsilon/(6R(r+1))$ suffices to correctly conclude either* OPT $< \alpha$ *or* OPT $\geq \alpha - \varepsilon$ *for the corresponding LP.*

Proof. Finding the optimal value λ^* of the game (5.6) up to additive error $\varepsilon^{(3)}$ will tell us at least one of the following two things:

- $\lambda^* > 0$
- $\lambda^* \leq 2\varepsilon^{(3)}$.

First assume we are in the case where $\lambda^* > 0$. In this case there is no $x^{(3)} \in \Delta^{n+2}$ such that

$$A^{(3)} x^{(3)} \leq 0.$$

On the other hand if we had OPT $\geq \alpha$, then there would be an x such that $Ax \leq b$, $\sum_i x_i \leq R, x \geq 0$ and $c^T x \geq \alpha$. For this x let

$$\hat{x} = \begin{pmatrix} x/(2R) \\ 1/2 - \sum_i x_i/(2R) \\ 1/2 \end{pmatrix}.$$

Then

$$A^{(3)}\hat{x} = \begin{pmatrix} 0 \\ 0 \\ (-c^T x + \alpha)/(2R) \\ (Ax - b)/(2R) \end{pmatrix} \leq 0,$$

which is a contradiction, hence OPT $< \alpha$.

Now we treat the other case: if $\lambda^* \leq 2\varepsilon^{(3)}$ and $x^* = (x, z, h)$ is a strategy with value λ^*, then we find that

$$A^{(3)} x^* \leq 2\varepsilon^{(3)}.$$

This implies that $|h - 1/2| \leq \varepsilon^{(3)}$. Since $-c^T x + h\alpha/R \leq 2\varepsilon^{(3)}$ we get that $c^T x \geq h\alpha/R - 2\varepsilon^{(3)} \geq \alpha/(2R) - 2(\alpha/(2R)+1)\varepsilon^{(3)}$. A similar argument also shows that for all $j \in [m]$

$$(Ax)_j \leq b_j/(2R) + 2(b_j/(2R)+1)\varepsilon^{(3)}.$$

Let $\hat{x} = 2Rx$, then

$$c^T\hat{x} \geq \alpha - 2(\alpha + 2R)\varepsilon^{(3)} \quad \geq \alpha - 6R\varepsilon^{(3)} \tag{5.7}$$

$$A\hat{x} \leq b + 2(b + 2R\mathbf{1})\varepsilon^{(3)} \leq b + 6R\varepsilon^{(3)}\mathbf{1}. \tag{5.8}$$

Let y^* be an optimal solution for the dual (5.5), such that $\sum_i y_i \leq r$. Then by applying weak duality on the relaxed constraints in (5.8) we find that

$$c^T\hat{x} \leq (b + 6R\varepsilon^{(3)}\mathbf{1})^T y^* \leq \mathrm{OPT} + 6Rr\varepsilon^{(3)},$$

and hence by (5.7) we can conclude

$$\mathrm{OPT} \geq \alpha - 6R(r+1)\varepsilon^{(3)} = \alpha - \varepsilon. \qquad \square$$

Via this reduction we give two new quantum LP-solvers. The first improves the error dependence of quantum LP-solvers to cubic; in contrast, an LP-solver obtained from our quantum SDP-solvers would have a fifth-power error dependence (see Section 4.8.4). The second solver is based on our sparse algorithm and is the first quantum LP-solver that depends on the sparsity of the LP instead of on n and m.

Theorem 5.13 *Given quantum query access to an LP of the form* (5.4)*, an ε-optimal and ε-feasible y can be found with probability $1-\delta$ using either*

- $\widetilde{\mathcal{O}}\big((\sqrt{n}+\sqrt{m})\gamma^3\big)$ *quantum queries to a dense matrix oracle and the same number of other gates, or*

- $\widetilde{\mathcal{O}}_{nm}\big(\sqrt{s}\gamma^{3.5}\big)$ *quantum queries to a sparse matrix oracle and the same number of other gates.*

where $\gamma = \frac{Rr}{\varepsilon}$.

Proof. The $\widetilde{\mathcal{O}}\big((\sqrt{n}+\sqrt{m})\gamma^3\big)$ bound follows directly from Lemma 5.12 and Theorem 5.8.

For the sparse case let s be the maximum of the sparsity of A, b and c. Then apart from the first two rows, every row and column of $A^{(1)}$ is $(s+3)$-sparse. However, the row sparsity only matters in the complexity of the Gibbs-sampling step, in which multiples of the all-one vector can be added to the exponent without changing the Gibbs state. Since the first two rows are the all-one vector plus a 1-sparse vector, we can treat them as effectively 1-sparse for the Gibbs-sampling step. The stated complexity bound then follows from Lemma 5.12 and Theorem 5.11. □

5.5 | An open question about non-negative LPs

The algorithm by Grigoriadis and Khachiyan was the starting point for a line of work about sublinear-time classical algorithms for linear programming [LN93; KY14]. These algorithms include so called *width-free non-negative LP-solvers.* Non-negative LPs are linear programs where all entries of A are non-negative, $b = \mathbf{1}$, and $c = \mathbf{1}$. An LP in primal form with these properties is called a *packing LP*, since for these kind of LPs we want to increase x as much as possible without violating constraints on the weighted sum of the entries. The dual of such an LP is called a *covering LP*. Packing and Covering LPs can be solved up to a *multiplicative error* in the optimal value without a dependence on R and r, making the algorithms a lot more practical then our general LP-solvers. It would be an interesting direction of research to see if these methods lend themselves to quantum speedups, possibly using the techniques discussed in this chapter. In Section 9.2.4 we will prove some negative results in this direction, but our results still leave some room for a quantum speedup.

Part II

Limitations of quantum computation

Chapter 6

Methods for lower bounding quantum query complexity

This chapter will introduce some basic concepts and results that will be used to prove lower bounds in the next chapters. In particular we will focus on lower bounding the quantum query complexity. We start by introducing the polynomial method as given by Beals et al. [BBCMW01], which allows us to lower bound the quantum query complexity of a function with the approximate degree of that function. After this, we will introduce the adversary bound as given by Ambainis [Amb00] and followup work by Høyer et al. [HLŠ07]. The adversary bound has many different forms, the most general of which can be written as an SDP and completely characterizes the quantum query complexity.

6.1 | Introduction

In Part 1 we stated algorithms in a *query model*: the input to an algorithm was given in the form of a unitary, called an *oracle*, that encodes the input. In complexity statements we counted both the number of queries to the oracle and the number of other two-qubit gates used. Since the query model hides the way the input is stored in the oracle, it allows us to focus on the structure of the algorithm. Furthermore, it allows for the input to be the output of another algorithm, where the query complexity gives the number of calls to this other algorithm.

The quantum query model also has an important application to lower bounds. While it is unclear how to lower bound the (quantum) time complexity directly, multiple ways of lower bounding the (quantum) query complexity are known. Since all queries take at least $\Omega(1)$ time, a lower bound on the query complexity also gives a lower bound on the time complexity. Of course these methods can never prove a lower bound on the time complexity that is larger than the input length, since the query complexity is always upper bounded by the input length. However, query lower bounds have been used to show the optimality of many quantum algorithms, including the Grover search algorithm.

Throughout this chapter we will use the problem of computing the OR-function, a decision version of the search problem, as an example:

> *Given an $x \in \{0,1\}^n$ such that $|x| = 0$ or $|x| = 1$, decide which of these cases holds using queries of the form $O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$ for $i \in [n]$ and $b \in \{0,1\}$.*

6.2 | The query model

In this section we will introduce a query-focused model of quantum computation. Let D be the set of labels for each possible input to a problem and let R be the set of possible outputs to this problem. For the search problem given above, $D = \big\{x \in \{0,1\}^n : |x| = 0 \text{ or } |x| = 1\big\}$ and $R = \{0,1\}$ (where the outcomes corresponds the possible Hamming weights of the input). Furthermore, let $\{O_x : x \in D\}$ be a set of unitaries, one for each possible input label. For the search problem these unitaries are the unitaries that act as $O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$.

We consider quantum algorithms that compute $f(x)$ for a function $f : D \to R$ using a number of applications of O_x and a number of x-independent gates. In addition to the qubits that O_x acts on, an algorithm might also use a workspace register.

Now assume we have a T-query algorithm and let us look at its general structure. All x-independent gates between the ith application of O_x and the $(i+1)$th application can be combined into a single unitary U_i that acts on all the qubits in the system. Additionally we write U_0 for the unitary consisting of the gates before the first query, and U_T for the unitary after the last query. Without loss of generality we may assume that our algorithm starts in the all-zero state, since any other state could be prepared by U_0, so after all the steps we end up with the state

$$U_T(O_x \otimes I)\dots(O_x \otimes I)U_1(O_x \otimes I)U_0|0\dots0\rangle.$$

We then measure this state in the computational basis. If the algorithm has error probability at most ρ, then the probability of measuring $f(x)$ in the first part of the state should be at least $1-\rho$ for all x:

$$\left\|\left(\left\langle f(x)\right| \otimes I\right)U_T(O_x \otimes I)\dots(O_x \otimes I)U_1(O_x \otimes I)U_0|0\dots0\rangle\right\|_2^2 \geq 1-\rho \text{ for all } x \in D. \tag{6.1}$$

The *query complexity* of f is the minimal query count T over all algorithms that compute f with the desired error probability. We write $Q_\rho(f)$ for the query complexity of f with error probability ρ. For error probability $1/3$ we simply write $Q(f)$.

6.3 | The polynomial method

The first method of lower bounding the quantum query complexity that was applicable to multiple problems was introduced by Beals et al. [BBCMW01] in 1998. They considered how the amplitudes of the state changed during the run of an algorithm and showed that the final measurement probability is a *multivariate polynomial* in terms of the input. For an algorithm which makes few queries to the input this polynomial will be of low degree. Hence, if we can show that all correct algorithms for a problem require high degree polynomials, then we can lower bound the query complexity. Unsurprisingly, their method is called *the polynomial method*.

For simplicity we will fix a finite set Σ, the elements of which are described by binary strings, and only consider decision problems for inputs $x \in \Sigma^n$ with queries of the form $O_x|i\rangle\left|y\right\rangle = |i\rangle\left|y \oplus x_i\right\rangle$ (where $\oplus$ denotes the bitwise XOR of two binary strings).

Lemma 6.1 : [BBCMW01] *Let Σ be a finite set and $f:\Sigma^n \to \{0,1\}$ be a decision problem. If O_x is of the form $O_x|i\rangle\left|y\right\rangle = |i\rangle\left|y \oplus x_i\right\rangle$, then the final acceptance probability of a T-query algorithm (the probability that the algorithm outputs a 1) is a polynomial of degree at most $2T$ in the variables $\delta_{x_i,y} := \left[x_i = y\right]$.*

Proof. Let $\mathscr{A}$ be a T-query quantum algorithm. Let s be the number of bits used to represent the elements of Σ and let m be the number of qubits used as additional workspace. Then $\mathscr{A}$ acts on the space $\mathbb{C}^n \otimes \mathbb{C}^{2^s} \otimes \mathbb{C}^{2^m}$. We will track the amplitudes of the state through the algorithm for an input x. Let $\alpha^{(t)}_{i,y,w}$ be the amplitude of the state $\left|i,y,w\right\rangle$ *before* U_t is applied. Let $\hat{\alpha}^{(t)}_{i,y,w}$ be the amplitude of the state $\left|i,y,w\right\rangle$ *after* U_t is applied. We will prove via induction on t that all the $\hat{\alpha}^{(t)}_{i,y,w}$ are polynomials of degree at most t in terms of the $\delta_{x_i,y}$.

Since the algorithm starts in the state $|0,0,0\rangle$, all $\alpha^{(0)}_{i,y,w}$ are 0 apart from $\alpha^{(0)}_{0,0,0}$ which is 1. Then $\hat{\alpha}^{(0)}_{i,y,w}$ is the (i,y,w) entry of the first column of U_0 and hence is of degree 0 in terms of the variables $\delta_{x_i,y}$.

Now assume $\hat{\alpha}^{(t)}_{i,y,w}$ is of degree at most t. Since $O_x|i\rangle\big|y\big\rangle|w\rangle = |i\rangle\big|y \oplus x_i\big\rangle|w\rangle$ we get $\alpha^{(t+1)}_{i,y\oplus x_i,w} = \hat{\alpha}^{(t)}_{i,y,w}$, or equivalently $\alpha^{(t+1)}_{i,y,w} = \hat{\alpha}^{(t)}_{i,y\oplus x_i,w}$. This last equation can be rewritten using the $\delta_{x_i,y}$ variables as

$$\alpha^{(t+1)}_{i,y,w} = \sum_{z\in\Sigma} \delta_{x_i,z}\hat{\alpha}^{(t)}_{i,y\oplus z,w}.$$

Since the $\hat{\alpha}^{(t)}$ are all of degree at most t, the $\alpha^{(t+1)}$ are of degree at most $t+1$ in terms of the $\delta_{x_i,y}$ variables. Because U_{t+1} is a linear map and independent of x, the $\hat{\alpha}^{(t+1)}$ are just linear combinations of the $\alpha^{(t+1)}$, and hence also of degree at most $t+1$.

Since

$$\|(\langle 1| \otimes I)U_T(O_x \otimes I)\ldots(O_x \otimes I)U_1(O_x \otimes I)U_0|0\ldots 0\rangle\|_2^2 = \sum_{y\in\Sigma, w\in[2^m]} \left|\hat{\alpha}^{(T)}_{1,y,w}\right|^2,$$

the final acceptance probability is a polynomial of degree at most $2T$ in terms of the $\delta_{x_i,y}$ variables. □

Lemma 6.1 naturally leads to a method for lower bounding the quantum query complexity of a function f: if a bounded-error algorithm $\mathscr{A}$ computes f on all inputs x, then the acceptance probability $P(x)$ of the algorithm should be close to $f(x)$ for all inputs. Hence, if all polynomials in the $\delta_{x_i,y}$ variables that approximate $f(x)$ are of a large degree, then the query complexity must be large as well.

Corollary 6.2 *Let Σ be a finite set, $f : \Sigma^n \to \{0,1\}$ be a decision problem, and O_x be of the form $O_x|i\rangle\big|y\big\rangle = |i\rangle\big|y \oplus x_i\big\rangle$. If every polynomial in terms of the $\delta_{x_i,y}$ variables that approximates $f(x)$ up to an additive error ρ has degree at least $2T$, then $Q_\rho(f) \geq T$.*

It might seem that this corollary would be hard to use since $f(x)$ is often stated in terms of the x_i and not the $\delta_{x_i,y}$ variables. However, in many cases the $\delta_{x_i,y}$ are quite natural. For example, if $\Sigma = \{0,1\}$, then $\delta_{x_i,1} = x_i$ and $\delta_{x_i,0} = 1 - x_i$. Hence for Boolean functions f it holds that approximate degree of f, i.e., the minimal degree of a polynomial that approximates f, lower bounds $2Q(f)$. In fact, the *exact* degree of f lower bounds the *exact* quantum query complexity as well. Recently Arunachalam, Briët and Palazuelos [ABP19] introduced a stricter notion of approximate degree that is exactly equal to the quantum query complexity.

Another obstacle in applying Corollary 6.2 might be that the multivariate degree can be hard to lower bound. However, as we will see in the next example, by averaging over certain sets of similar inputs we will often be able to only consider univariate polynomials.

An example: the OR function [BBCMW01] Let $f : \big\{x \in \{0,1\}^n : |x| = 0 \text{ or } |x| = 1\big\} \to \{0,1\}$ be the OR function restricted to x with Hamming weight 0 or 1, that is, $f(0^n) = 0$ and $f(x) = 1$ for all x with Hamming weight 1. Let $\mathscr{A}$ be a T-query bounded-error algorithm that computes f, and let $p(x)$ be the degree-$2T$ polynomial that gives its acceptance probability. Since $\mathscr{A}$ is a correct algorithm, $p(0^n) \leq 1/3$ and $p(x) \geq 2/3$ if

$|x| = 1$. The algorithm will still produce some probability distribution when an oracle O_x is given for an $x \in \{0,1\}^n$ outside the domain, therefore $p(x) \in [0,1]$ for all $x \in \{0,1\}^n$.

Since $f(x)$ does not change when we permute the entries of x, we know that the average

$$q(x) = \frac{1}{n!} \sum_{\pi \in S_n} p(\pi(x))$$

still approximates f and still has degree $2T$. Note that way me assume that q is multilinear since $x_i^2 = x_i$ for $x_i \in \{0,1\}$. In q all degree-d monomials have the same coefficient, which we will denote by α_d. We rewrite q as

$$q(x) = \sum_{d=0}^{2T} \alpha_d \sum_{S \subseteq [n]:|S|=d} \prod_{i \in S} x_i = \sum_{d=0}^{2T} \alpha_d \sum_{S \subseteq [n]:|S|=d} [\forall i \in S : x_i = 1].$$

On input x there are $\binom{|x|}{d}$ terms in $\sum_{S \subseteq [n]:|S|=d} [\forall i \in S : x_i = 1]$ that are one, and all other terms are zero. This shows that q only depends on $|x|$, hence we can define

$$r(|x|) := q(x) = \sum_{d=0}^{2T} \alpha_d \binom{|x|}{d}.$$

Since $\binom{|x|}{d}$ is a degree-d polynomial in $|x|$, $r(|x|)$ is an degree-$2T$ polynomial. Note that while $q(x)$ was multilinear, r is a higher degree polynomial in a single variable.

We know that $r(0) \leq 1/3$, $r(1) \geq 2/3$ and that $r(w) \in [0,1]$ for all $w \in [n]$. Classical approximation theory now tells us that $\deg(r) \geq \Omega(\sqrt{n})$ [RC66; EZ64]. Hence $T \geq \Omega(\sqrt{n})$.

6.4 | The adversary method

In 2000 Ambainis introduced an alternative method for proving quantum query lower bounds, the *adversary method* [Amb00]. This method considers how well an algorithm can distinguish inputs that should lead to different outputs. In particular, the unitary U_t matrices do not change the angle between states, so if the final states the algorithm ends with for two inputs x and y are very different, then the oracle queries must have introduced this difference. The adversary method gives a method for showing that this cannot be the case for all input pairs without using a large number of queries. Often the adversary method is easier to apply than the polynomial method, however, the adversary method is not capable of lower bounding the *exact* quantum query complexity.

There are many different versions of the adversary method known [ŠS05]. In the next section we will introduce the most basic version without a proof. In Section 6.4.2 we will prove a version of the adversary method that works for phase oracles $|x\rangle \mapsto e^{\mathbf{i}z(x)}|x\rangle$ where $z : D \to \mathbb{R}$ is a function. Finally we will introduce the general adversary method due to Høyer, Lee and Špalek [HLŠ07]. This version of the adversary method can be written as an SDP and fully characterizes the bounded error quantum query complexity (up to a constant).

Many other versions of the adversary method are known. The most complete characterization is due to Belovs in [Bel15]. In his work tight characterizations in almost any quantum query setting are given. However, the resulting optimization problems are very complex and hard to work with.

6.4.1 | The basic adversary method

In its most basic version the adversary lower bound can be stated as follows:

Lemma 6.3 : The basic adversary method *[Amb00] Let Σ be a finite set, $D \subseteq \Sigma^n$, $f : D \to \{0,1\}$ be a decision problem, and O_x be of the form $O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$. Let $X \subseteq f^{-1}(0)$ and $Y \subseteq f^{-1}(1)$. Furthermore let $R \subseteq X \times Y$ be a binary relation on $X \times Y$. For $x \in X$, we write $R(x) = \{y \in Y : (x,y) \in R\}$ (and similar for $y \in Y$). Define*

$$\begin{aligned}
m_0 &:= \min_{x\in X} |R(x)| \\
m_1 &:= \min_{y\in Y} |R(y)| \\
\ell_0 &:= \max_{i\in[n],x\in X} \sum_{y\in R(x)} [x_i \neq y_i] \\
\ell_1 &:= \max_{i\in[n],y\in Y} \sum_{x\in R(y)} [x_i \neq y_i].
\end{aligned}$$

Then

$$Q_{1/3}(f) \geq \Omega\left(\sqrt{\frac{m_0 m_1}{\ell_0 \ell_1}}\right).$$

Using the adversary method comes down to choosing appropriate X, Y and R, and proving bounds on m_0, m_1, ℓ_0 and ℓ_1. As an example we again use the OR function.

An example: the OR function [Amb00] Let $f : \{x \in \{0,1\}^n : |x| = 0 \text{ or } |x| = 1\} \to \{0,1\}$ be the OR function. Let $X = f^{-1}(0) = \{0^n\}$ and $Y = f^{-1}(1) = \{y \in \{0,1\}^n : |y| = 1\}$. Take $R = X \times Y$. Then:

$$\begin{aligned}
m_0 &= \min_{x\in X} |R(x)| = n \\
m_1 &= \min_{y\in Y} |R(y)| = 1 \\
\ell_0 &= \max_{i\in[n],x\in X} \sum_{y\in R(x)} [x_i \neq y_i] = 1 \\
\ell_1 &= \max_{i\in[n],y\in Y} \sum_{x\in R(y)} [x_i \neq y_i] = 1,
\end{aligned}$$

and hence

$$Q_{1/3}(f) \geq \Omega(\sqrt{n}).$$

6.4.2 | The adversary method for phase oracles

In this section we consider problems for which the input is a function from $x : Q \to \mathbb{R}$ where Q is a finite set of possible queries. We assume that x can be accessed via queries $O_x|q\rangle = e^{\mathbf{i}x(q)}|q\rangle$. We consider a finite subset D of all possible such functions, and are asked to solve some decision problem $f : D \to \{0,1\}$. For example, for Boolean decision problems on n bits we would have $Q = [n]$, $D \subseteq \{0,1\}^n$, and we would identify a string $x \in D$ with a function $x : [n] \to \{0,1\}$ that on input i returns the ith entry in x.

Even though this type of queries are often used for accessing a binary string (see for example Section 2.2.6) and are used in quantum gradient computation (see Section 2.3.3), we are not aware of any prior direct proof of the adversary method for this setting. It is possible to obtain the following lemma as a corollary of the results in [Bel15], but due to the complex nature of the results in [Bel15] a direct proof is easier and clearer. The proof is very similar to the proof of the basic adversary method from [Amb00].

Lemma 6.4 : The phase adversary method *Let Q be a finite set, and let D be a finite set of functions from Q to $\mathbb{R}$. Let $f : D \to \{0,1\}$ be a decision problem on this set of functions, and O_x for $x \in D$ be of the form $O_x|q\rangle = e^{\mathbf{i}x(q)}|q\rangle$. Let $X \subseteq f^{-1}(0)$ and $Y \subseteq f^{-1}(1)$. Furthermore let $R \subseteq X \times Y$ be a binary relation on $X \times Y$. For $x \in X$, we write $R(x) = \{y \in Y : (x,y) \in R\}$ (and similar for $y \in Y$). Define*

$$\begin{aligned} m_0 &:= \min_{x\in X} |R(x)| \\ m_1 &:= \min_{y\in Y} |R(y)| \\ \ell_{q,x} &:= \sum_{y\in R(x)} |x(q)-y(q)| \\ \ell_{q,y} &:= \sum_{x\in R(y)} |x(q)-y(q)| \\ \ell_{\max} &:= \max_{q\in Q, x\in X, y\in Y} \ell_{q,x}\ell_{q,y}. \end{aligned}$$

Then

$$Q_{1/3}(f) \geq \Omega\left(\sqrt{\frac{m_0 m_1}{\ell_{\max}}}\right).$$

Proof. Assume there is an algorithm that makes T queries to the input. Let $|\psi_x^t\rangle$ be the state we get by running the algorithm for input x up to and including the tth query. We define the progress measure $\Phi(t) := \sum_{(x,y)\in R} \left|\left\langle \psi_x^t \middle| \psi_y^t \right\rangle\right|$. Notice that $\Phi(0) = |R|$ and since the algorithm has to distinguish all pairs in R with bounded error probability we have $\Phi(T) \leq 0.99|R|$. To complete the proof we will show that the progress matter does not change too much in each step:

$$\Phi(t-1) - \Phi(t) \leq 2\sqrt{\frac{\ell_{\max}}{m_0 m_1}}|R|.$$

First, since all the U_t are unitary and input-independent, the progress measure can only change in the query steps. Now, consider a particular x, y and t. Write $\left|\overline{\psi}_x^t\right\rangle$ for the state on input x, just before the tth query, and similarly for y. We can write

$$\left|\overline{\psi}_x^t\right\rangle = \sum_{q\in Q} \alpha_q |q\rangle \otimes |\phi_q\rangle$$

and

$$\left|\overline{\psi}_y^t\right\rangle = \sum_{q\in Q} \beta_q |q\rangle \otimes |\xi_q\rangle.$$

Hence after the query we get

$$|\psi_x^t\rangle = \sum_{q\in Q} e^{\mathbf{i}x(q)} \alpha_q |q\rangle \otimes |\phi_q\rangle$$

and

$$\left|\psi_y^t\right\rangle = \sum_{q\in Q} e^{\mathbf{i}y(q)} \beta_q |q\rangle \otimes |\xi_q\rangle.$$

So the absolute change in their inner product is:

$$\begin{aligned}\left|\left\langle\overline{\psi}_x^t\middle|\overline{\psi}_y^t\right\rangle - \left\langle\psi_x^t\middle|\psi_y^t\right\rangle\right| &\leq \left|\sum_{q\in Q} \alpha_q \beta_q^* \langle\phi_q|\xi_q\rangle\left(1 - e^{\mathbf{i}(x(q)-y(q))}\right)\right| \\ &\leq \sum_{q\in Q} |\alpha_q| \cdot |\beta_q| \cdot |\langle\phi_q|\xi_q\rangle| \cdot |x(q) - y(q)| \\ &\leq \sum_{q\in Q} |\alpha_q| \cdot |\beta_q| \cdot |x(q) - y(q)|,\end{aligned}$$

where we used that $\left|1 - e^{\mathbf{i}\theta}\right| \leq |\theta|$. Now let $x_q = \operatorname{argmax}_{x\in X} \ell_{q,x}$ and $y_q = \operatorname{argmax}_{y\in Y} \ell_{q,y}$. We can now analyze the difference in the progress measure:

$$\begin{aligned}|\Phi(t) - \Phi(t-1)| &\leq \sum_{(x,y)\in R} \sum_{q\in Q} |\alpha_q| \cdot |\beta_q^*| \cdot |x(q) - y(q)| \\ &= \sum_{q\in Q} \sum_{(x,y)\in R} |x(q) - y(q)| \sqrt{|\alpha_q|^2 \sqrt{\frac{m_0 \ell_{q,y_q}}{m_1 \ell_{q,x_q}}} |\beta_q^*|^2 \sqrt{\frac{m_1 \ell_{q,x_q}}{m_0 \ell_{q,y_q}}}} \\ &\leq \frac{1}{2} \sum_{q\in Q} \sum_{(x,y)\in R} |x(q) - y(q)| \left(|\alpha_q|^2 \sqrt{\frac{m_0 \ell_{q,y_q}}{m_1 \ell_{q,x_q}}} + |\beta_q^*|^2 \sqrt{\frac{m_1 \ell_{q,x_q}}{m_0 \ell_{q,y_q}}}\right),\end{aligned}$$

where we used the arithmetic mean geometric mean inequality. Let us look at the two terms separately:

$$
\begin{aligned}
&\frac{1}{2}\sum_{q\in Q}\sum_{(x,y)\in R}|x(q)-y(q)||\alpha_q|^2\sqrt{\frac{m_0\ell_{q,y_q}}{m_1\ell_{q,x_q}}}\\
&\quad=\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sum_{q\in Q}|\alpha_q|^2\sqrt{\frac{\ell_{q,y_q}}{\ell_{q,x_q}}}\sum_{x\in X}\sum_{y\in R(x)}|x(q)-y(q)|\\
&\quad=\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sum_{q\in Q}|\alpha_q|^2\sqrt{\frac{\ell_{q,y_q}}{\ell_{q,z_q}}}\sum_{x\in X}\ell_{q,x}\\
&\quad\leq\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sum_{q\in Q}|\alpha_q|^2\sqrt{\frac{\ell_{q,y_q}}{\ell_{q,x_q}}\ell_{q,x_q}|X|}\\
&\quad=\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sum_{q\in Q}|\alpha_q|^2\sqrt{\ell_{q,y_q}\ell_{q,x_q}}|X|\\
&\quad\leq\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sqrt{\ell_{\max}}|X|\sum_{q\in Q}|\alpha_q|^2\\
&\quad\leq\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sqrt{\ell_{\max}}|X|\\
&\quad\leq\frac{1}{2}\sqrt{\frac{m_0}{m_1}}\sqrt{\ell_{\max}}\frac{|R|}{m_0}\\
&\quad=\frac{1}{2}\sqrt{\frac{\ell_{\max}}{m_0m_1}}|R|.
\end{aligned}
$$

By symmetry we have the same bound for the other term. We conclude that

$$
|\Phi(t)-\Phi(t-1)|\leq\sqrt{\frac{\ell_{\max}}{m_0m_1}}|R|,
$$

which concludes the proof. □

It can be shown that Lemma 6.3 follows from Lemma 6.4. We will not give an exact proof, but the reasoning is roughly as follows. It suffices to show that there is a phase oracle that can be used to implement the oracle from Lemma 6.3 and that for this phase oracle $\ell_{\max}$ is not bigger than the $\ell_0\ell_1$ from Lemma 6.3. Without loss of generality assume that $\Sigma=[k]$ for some positive integer k. Now for a $\hat{x}\in[k]^n$, let

$$
x(i,c)=2\pi\hat{x}_ic/k \mod 2\pi
$$

be a function from $[n]\times[k]$ to the reals.

To implement a query of the form $|i,0\rangle\mapsto|i,\hat{x}_i\rangle$ using phase queries to x, set up the superposition $|i\rangle\sum_{c\in[k]}|c\rangle$, query O_x to get $|i\rangle\sum_{c\in[k]}e^{\mathbf{i}2\pi\hat{x}_ic/k}|c\rangle$ and then apply the inverse quantum Fourier transform to the second register to get $|i,\hat{x}_i\rangle$. This allows us to implement the queries from Lemma 6.3 using phase queries to the x functions and hence the phase queries are at least as strong as the queries from Lemma 6.3. Note that if $\hat{x}_i = \hat{y}_i$, then $|x(i,c) - y(i,c)| = 0$ and that for all i and c the difference $|x(i,c) - y(i,c)| \leq 2\pi$. It follows that the parameter $\ell_{\max}$ of the phase adversary bound is upper bounded by the quantity $\mathcal{O}(\ell_0 \ell_1)$ from the basic adversary bound.

Example : Ground state energy finding using Hamiltonian evolution As an example of how the phase adversary method can be applied, consider the following problem: let H be a Hamiltonian with $\|H\| \leq 1$ acting on n qubits, accessed by queries of the form $O_H = e^{\mathbf{i}H}$, determine whether the ground state energy $\lambda_{\min}(H)$ is 0 or larger than ε.

To prove a lower bound on this problem, let $N = 2^n$ and consider the sets $X = \{\varepsilon I\}$ and $Y = \{\varepsilon(I - E_{ii}) : i \in [N]\}$ of $N \times N$ matrices. Let $R = X \times Y$. All matrices are diagonal, so the unitaries corresponding to these Hamiltonians correspond to phase queries. The matrix εI corresponds to the constant function $x(q) = \varepsilon$, and the matrices $\varepsilon(I - E_{ii})$ correspond to functions that are ε everywhere except at input i, where they are zero.

Considering the inputs as functions we get

$$\begin{aligned}
m_0 &= \min_{x \in X} |R(x)| = N \\
m_1 &= \min_{y \in Y} |R(y)| = 1 \\
\ell_{q,x} &= \sum_{y \in R(x)} |x(q) - y(q)| = \varepsilon \\
\ell_{q,y} &= \sum_{x \in R(y)} |x(q) - y(q)| \leq \varepsilon \\
\ell_{\max} &= \max_{q \in Q, x \in X, y \in Y} \ell_{q,x} \ell_{q,y} \leq \varepsilon^2,
\end{aligned}$$

and hence the quantum query complexity is at least

$$\Omega\left(\frac{\sqrt{N}}{\varepsilon}\right).$$

In fact, this matches the upper bound that can be achieved using Lemma 4.9.

6.4.3 | The general adversary method

We finish this chapter by stating the general adversary method, which was first introduced in [HLŠ07]. This version of the adversary method takes the form of an SDP. It has been shown for Boolean functions that a feasible point for the dual of this SDP can be turned into a quantum algorithm that uses a number of queries equal to the objective value [Rei11] (up to a multiplicative constant). This implies that the general adversary method is actually tight and completely characterizes the bounded-error quantum query complexity up to a multiplicative constant.

Lemma 6.5 : The general adversary method *[HLŠ07] Let Q be a finite set of possible queries, $f : D \to R$ be a query problem, and O_x be of the form $O_x|q\rangle|b\rangle = |i\rangle|b \oplus x(q)\rangle$ where $x(q)$ is the result of query q on input x. We call $\Gamma \in \mathbb{R}^{D\times D}$ an adversary matrix if Γ is not zero and for $x, y \in D$ with $f(x) = f(y)$ we have $\Gamma_{x,y} = 0$. Let G be the set of all adversary matrices and define Δ_q by $(\Delta_q)_{x,y} := [x(q) \neq y(q)]$. Let*

$$\mathrm{ADV}^{\pm}(f) = \max_{\Gamma\in G} \frac{\|\Gamma\|}{\max_{q\in Q}\|\Gamma \circ \Delta_q\|}.$$

Then

$$Q_{1/3}(f) \geq \Omega\big(\mathrm{ADV}^{\pm}(f)\big).$$

Chapter 7

Lower bounds on the query complexity of convex optimization

In Chapter 3 we introduced five black-box oracles for convex optimization and we gave a quantum algorithm for efficiently implementing a SEP oracle using a MEM oracle. In this chapter we will finish our discussion of black-box oracles for convex optimization by proving query lower bounds on reductions between those oracles. We first prove an $\Omega(n)$ *classical* query lower bound on implementing a SEP oracle using a MEM oracle, which shows that the result from Chapter 3 is indeed an exponential speedup. After this we consider two settings for implementing OPT using SEP queries, while knowing an interior point and while not knowing an interior point of the convex set. We prove $\Omega(\sqrt{n})$ and $\Omega(n)$ *quantum* query lower bounds respectively for these settings.

This chapter is based on Section 5 of the paper *Convex optimization using quantum oracles* by J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf [AGGW18].

7.1 | Introduction

For a convex set K satisfying $B(0,r) \subseteq K \subseteq B(0,R)$, we have shown in Theorem 3.18 that one can implement a SEP(K) oracle with $\widetilde{\mathcal{O}}_{\frac{Rn}{r\varepsilon}}(1)$ quantum queries to a MEM(K) oracle if the membership oracle is sufficiently precise. In this chapter we start by showing that this is exponentially better than what can be achieved using classical access to a membership oracle. We also investigate how many queries to a membership/separation oracle are needed in order to implement an optimization oracle. Our results are as follows (see also Figure 3.1):

- We show that $\Omega(n)$ classical queries to a membership oracle are needed to implement a weak separation oracle.
- We show that $\Omega(n)$ classical (respectively $\Omega(\sqrt{n})$ quantum) queries to a separation oracle are needed to implement a weak optimization oracle; even when we *know an interior point* in the set.
- We show an $\Omega(n)$ lower bound on the number of classical and/or quantum queries to a separation oracle needed to optimize over the set when we *do not know an interior point.*

Although these results imply that there is no quantum speedup for finding a point in the interior of K using separation queries, they do leave open the possibility for a quadratic speedup for optimization when an interior point is known in the convex set. We view the closing of the gap between the upper and lower bound for this problem as an important direction for future work. As we describe in Section 7.5 one possible approach that could lead to an $\Omega(n)$ lower bound is lower bounding the number of matrix-vector queries $x \mapsto Ax$ needed to determine the rank of a matrix $A \in \mathbb{R}^{n\times n}$. In Chapter 8 we will consider this problem over $\mathbb{F}_p$ instead of $\mathbb{R}$.

The same polarity argument as in Section 3.5 shows that algorithms for optimization using separation are essentially equivalent to algorithms for separation using optimization. This turns our lower bound on the number of separation queries needed to implement an optimization oracle into a lower bound on the reverse direction.

In this chapter we will always assume that the input oracle is a strong oracle but the output oracle is allowed to be a weak oracle with error ε. Furthermore, we will make sure that R, $1/r$, and $1/\varepsilon$ are all upper bounded by a polynomial in n. This guarantees that the lower bound is based on the dimension of the problem, not the required precision.

7.2 | Classical lower bound on SEP using MEM

Here we show that a separation query can provide $\Omega(n)$ bits of information about the underlying convex set K; since a classical membership query returns a 0 or a 1 and hence can give at most 1 bit of information[1], this theorem immediately implies a lower bound of $\Omega(n)$ on the number of classical membership queries needed to implement one separation query.

Theorem 7.1 *Let $\varepsilon \le \frac{39}{1600}$. There exist a set of $m = 2^{\Omega(n)}$ convex sets $K_1, \dots, K_m \subseteq \mathbb{R}^n$ and points $y, x_0 \in \mathbb{R}^n$ such that $B(x_0, 1/3) \subseteq K_i \subseteq B\big(x_0, 2\sqrt{n}\big)$ for all $i \in [m]$, and such that $\mathrm{SEP}_{\varepsilon,0}(K_i)$ and $\mathrm{SEP}_{\varepsilon,0}(K_j)$ cannot return the same result when queried with the point y, unless $i = j$.*

Proof. Let $h_1, \dots, h_m \in \mathbb{R}^n$ be a set of $m = 2^{\Omega(n)}$ entrywise non-negative unit vectors such that $\langle h_i, h_j \rangle \le 0.51$ for all distinct $i, j \in [m]$.[2]

Now pick an $i \in [m]$ and define $\hat{K}_i := \{x : \langle h_i, x\rangle \le 0\} \cap B\big(0, \sqrt{n}\big)$ and $K_i := B\big(\hat{K}_i, \varepsilon\big)$. Then $\hat{K}_i = B(K_i, -\varepsilon)$. Note that for $x_0 = -\mathbf{1}/3$ we have $B(x_0, 1/3) \subseteq K_i \subseteq B\big(x_0, 2\sqrt{n}\big)$. We claim that a query to $\mathrm{SEP}_{\varepsilon,0}(K_i)$ with the point $y = 3\varepsilon\mathbf{1} \in \mathbb{R}^n$ will identify h_i. In particular, we will show that $\mathrm{SEP}_{\varepsilon,0}(K_i)$ will return a vector g that has a large inner product with h_i and a small inner product with all other h_j.

First note that $y \notin B(K_i, \varepsilon)$, since $\hat{K}_i$ does not contain any entrywise positive vectors and y has distance at least 3ε from all vectors that have at least one non-positive entry. Hence a separation query with y will return a unit vector g such that for all $x \in \hat{K}_i$

$$\langle g, x \rangle \le \langle g, y \rangle + \varepsilon \le \|g\| \cdot \|y\| + \varepsilon \le \big(3\sqrt{n} + 1\big)\varepsilon \le 4\sqrt{n}\varepsilon. \tag{7.1}$$

Now consider the specific point x that is the projection of g onto $h_i^\perp$ (the hyperplane orthogonal to h_i) scaled by a factor $\sqrt{n}$, i.e., $x = \sqrt{n}\big(g - \langle g, h_i\rangle h_i\big)$. Since $\langle h_i, x\rangle = 0$ and $\|x\| \le \sqrt{n}$, we have $x \in \hat{K}_i$. Therefore (7.1) gives the following inequality

$$\sqrt{n}\Big(1 - \langle g, h_i\rangle^2\Big) = \langle g, x\rangle \le 4\sqrt{n}\varepsilon.$$

Hence $\big|\langle g, h_i\rangle\big| \ge \sqrt{1 - 4\varepsilon} \ge \frac{19}{20}$. This implies that $g - h_i$ or $g + h_i$ has length at most $\sqrt{2 - 2\big|\langle g, h_i\rangle\big|} \le \sqrt{\frac{1}{10}}$; assume the former for simplicity. Now for all $j \ne i$ we have

$$\big|\langle g, h_j\rangle\big| \le \big|\langle g - h_i, h_j\rangle\big| + \big|\langle h_i, h_j\rangle\big| \le \sqrt{\frac{1}{10}} + 0.51 < \frac{9}{10}.$$

Hence g can only be returned by $\mathrm{SEP}_{\varepsilon,0}(K_i)$. □

[1] This is not true for *quantum* membership queries!

[2] We can show that such a set of vectors exists as follows. Let $n = ck$ for sufficiently large constant c. Choose $m = 2^k$ (which is $2^{\Omega(n)}$) uniformly random vectors $v_1, \dots, v_m$ in $\{0,1\}^n$. Note that the expected Hamming weight of one such vector is $n/2$, and the expected inner product between two such vectors is $n/4$ (the inner product just counts for how many of the n bit-positions both vectors have a 1). By a standard calculation (Chernoff bound plus a union bound), one can show that with high probability these 2^k vectors each have Hamming weight $\ge 0.495n$, and the inner product between any two of them is $\le 0.252n$. Fix 2^k such vectors with these properties, and define $h_i := v_i/\|v_i\|$. These are unit vectors with non-negative entries, and pairwise inner products $\langle h_i, h_j\rangle = \langle v_i, v_j\rangle/(\|v_i\|\|v_j\|) \le 0.252n/(0.495n) \le 0.51$.

If we would be given one of the $m = 2^{\Omega(n)}$ convex sets uniformly at random, then a single separation query would allow us to learn $\Omega(n)$ bits of information about our input. However, a classical membership query can teach us only a single bit of information about our input. It follows that $\Omega(n)$ classical membership queries are necessary to implement a single separation query.

7.3 | Query lower bound for OPT using SEP with an interior point

We now consider lower bounding the number of quantum queries to a separation oracle needed to do optimization. In fact, we prove a lower bound on the number of separation queries needed for validity, which implies the same bound on optimization. We will use a reduction from a version of the *search* problem:

Given $z \in \{0,1\}^n$ such that either $|z| = 0$ or $|z| = 1$, decide which of the two holds.

It is not hard to see that if the access to z is given via classical queries, then $\Omega(n)$ queries are needed. As seen in Chapter 6, $\Omega(\sqrt{n})$ queries are needed if we allow quantum queries. We use this problem to show that there exist convex sets for which it is hard to construct a weak validity oracle, given a strong separation oracle. Since a separation oracle can be used as a membership oracle, this gives the same hardness result for constructing a weak validity oracle from a strong membership oracle.

Theorem 7.2 *Let $0 < \rho \leq 1/3$. Let $\mathscr{A}$ be an algorithm that implements a $\mathrm{VAL}_{(5n)^{-1},\rho}(K)$ oracle for every convex set K (with $B(x_0, r) \subseteq K \subseteq B(x_0, R)$) using only queries to a $\mathrm{SEP}_{0,0}(K)$ oracle, and unitaries that are independent of K. Then the following statements are true, even when we restrict to convex sets K with $r = 1/3$ and $R = 2\sqrt{n}$:*

- *if the queries to $\mathrm{SEP}_{0,0}(K)$ are classical, then the algorithm uses $\Omega(n)$ queries.*
- *if the queries to $\mathrm{SEP}_{0,0}(K)$ are quantum, then the algorithm uses $\Omega(\sqrt{n})$ queries.*

Proof. Let $z \in \{0,1\}^n$ have Hamming weight $|z| = 0$ or $|z| = 1$. We construct a set K_z in such a way that solving the weak validity problem solves the search problem for z, while separation queries for K_z can be answered using a single query to z. The known classical and quantum lower bounds on the search problem then imply the two claims of the theorem, respectively.

Define $K_z := \prod_{i=1}^{n}[-1, z_i]$. Observe that if we set $x_0 = (-1/2, \ldots, -1/2)$, then $B\left(x_0, \frac{1}{3}\right) \subseteq K_z \subseteq B\left(x_0, 2\sqrt{n}\right)$.

We first show how to implement a strong separation oracle using a single query to z. Suppose the input is the point y. The strong separation oracle works as follows:

1. If $y \in [-1,0]^n$, then return the statement that $y \in B(K_z, 0) = K_z$.

2. If $y \not\in [-1,1]^n$, then return a hyperplane that separates y from $[-1,1]^n$ (and hence from K_z).

3. Let i be such that $y_i > 0$. Query z_i.

   (a) If $z_i = 1$ and i is the only index such that $y_i > 0$, then return that $y \in B(K_z, 0) = K_z$.

   (b) If $z_i = 1$ and there is a $j \neq i$ such that $y_j > 0$, return the separating hyperplane corresponding to $x_j \leq y_j$.

   (c) If $z_i = 0$, then return the separating hyperplane corresponding to $x_i \leq y_i$.

We show that a validity query over K_z with the direction $c = \frac{1}{\sqrt{n}}(1,\ldots,1) \in \mathbb{R}^n$, value $\gamma = \frac{1}{2\sqrt{n}}$ and error $\varepsilon = \frac{1}{5n}$ solves the search problem:

- If $|z| = 0$, then for all points $x \in K_0$ we have $\langle c, x\rangle \leq 0$. Thus, for all points $x \in B(K_0, \varepsilon)$ we have $\langle c, x\rangle \leq \varepsilon < \gamma - \varepsilon$. Hence validity will have to return that $\langle c, x\rangle \leq \gamma + \varepsilon$ holds for all $x \in B(K_0, -\varepsilon)$ since the other possible output is not true.

- If $|z| = 1$, then the point $z \in K_z$ satisfies $\langle z, c\rangle = \frac{1}{\sqrt{n}}$ and therefore $x = z - \varepsilon e \in B(K_z, -\varepsilon)$ satisfies $\langle c, x\rangle = \frac{1}{\sqrt{n}} - \sqrt{n}\varepsilon > \gamma + \varepsilon$. Hence validity will have to return that $\langle c, x\rangle \geq \gamma - \varepsilon$ holds for some $x \in B(K_z, \varepsilon)$ since the other possible output is not true.

□

7.4 | Query lower bound for OPT using SEP without an interior point

We now lower bound the number of quantum queries to a separation oracle needed to solve the optimization problem, if our algorithm does not already know an interior point of K. In fact we prove a lower bound on finding a point close to K using separation queries, which implies the lower bound on the number of separation queries needed for optimization since OPT returns a point close to the set.

We prove our lower bound by a reduction to the problem of learning z with *first-difference queries*:

> *Let $z \in \{0,1\}^n$ be a unknown binary string. For a given guess $g \in \{0,1\}^n$ a query returns the first index in $[n]$ for which the binary strings z and g differ (or it returns $n+1$ if $z = g$). Recover the whole of z.*

First we prove an $\Omega(n)$ quantum query lower bound for this problem.[3]

[3] Note that this is a strengthening of the $\Omega(n)$ quantum query lower bound for binary search on a space of size 2^n by Ambainis [Amb99], since first-difference queries are at least as strong as the queries one makes in binary search.

Theorem 7.3 *Let $z \in \{0,1\}^n$ be an unknown string accessible by an oracle acting as $O_z|g,b\rangle = |g, b \oplus f(g,z)\rangle$, where $f(g,z)$ is the first index for which z and g differ, more precisely $f(g,z) = \min\{i \in [n] : g_i \neq z_i\}$ if $g \neq z$ and $f(g,z) = n+1$ otherwise. Then every bounded-error quantum algorithm that outputs z uses at least $\Omega(n)$ queries to O_z.*

Proof. We will use the general adversary bound [HLŠ07] as introduced in Section 6.4.3 in Lemma 6.5. For this problem all inputs require different outputs, hence $\Gamma \in \mathbb{R}^{2^n \times 2^n}$ is an *adversary matrix* if it is a non-zero matrix with zero diagonal whose rows and columns are indexed by all $z \in \{0,1\}^n$. For $g \in \{0,1\}^n$, the $\Delta_g \in \{0,1\}^{2^n \times 2^n}$ matrix is such that the (z,z') entry of Δ_g is 0 if and only if $f(g,z) = f(g,z')$.

We claim that Lemma 6.5 gives a lower bound of $\Omega(n)$ for the adversary matrix Γ defined as

$$\Gamma_{z,z'} = \begin{cases} 2^{f(z,z')} & \text{if } z \neq z' \\ 0 & \text{if } z = z'. \end{cases}$$

It is easy to see that Γ is indeed an adversary matrix since it is zero on the diagonal and non-zero everywhere else. Furthermore, the all-one vector e is an eigenvector of Γ with eigenvalue $n2^n$:

$$\begin{aligned}(\Gamma e)_z &= \sum_{z' \in \{0,1\}^n} \Gamma_{z,z'} \\ &= \sum_{d=1}^{n} 2^d \cdot |\{z' \in \{0,1\}^n : f(z,z') = d\}| \\ &= \sum_{d=1}^{n} 2^d 2^{n-d} \\ &= n2^n.\end{aligned}$$

So $\Gamma e = n2^n e$ and hence $\|\Gamma\| \geq n2^n$. It remains to upper bound the operator norm of the $\Gamma \circ \Delta_g$ matrices. From the definition of Δ_g it follows that

$$\left(\Gamma \circ \Delta_g\right)_{z,z'} = 2^{f(z,z')}\left[f(g,z) \neq f(g,z')\right].$$

Let $\Gamma_g := \Gamma \circ \Delta_g$. We will show an upper bound on $\left\|\Gamma_g\right\|$. We decompose Γ_g in an "upper-triangular" and a "lower-triangular" part:

$$\begin{aligned}\left(\Gamma_g^U\right)_{z,z'} &:= 2^{f(z,z')}\left[f(g,z) < f(g,z')\right] = 2^{f(g,z)}\left[f(g,z) < f(g,z')\right] \\ \left(\Gamma_g^L\right)_{z,z'} &:= 2^{f(z,z')}\left[f(g,z') < f(g,z)\right] = 2^{f(g,z')}\left[f(g,z') < f(g,z)\right].\end{aligned} \tag{7.2}$$

So $\Gamma_g = \Gamma_g^U + \Gamma_g^L$ and $\Gamma_g^U = \left(\Gamma_g^L\right)^T$. Hence by the triangle inequality we have

$$\left\|\Gamma_g\right\| \leq \left\|\Gamma_g^U\right\| + \left\|\Gamma_g^L\right\| = 2\left\|\Gamma_g^U\right\|. \tag{7.3}$$

It thus suffices to upper bound $\left\|\Gamma_g^U\right\|$. Notice that as (7.2) shows, $\left(\Gamma_g^U\right)_{z,z'}$ only depends on the values $f(g,z)$, $f(g,z')$. Since the range of $f(g,\cdot)$ is $[n+1]$, we can think of Γ_g^U as an $(n+1)\times(n+1)$ block-matrix, where the blocks are determined by the values of $f(g,z)$ and $f(g,z')$, and within a block all matrix elements are the same. Also observe that for all $k\in[n]$ there are 2^{n-k} bitstrings $y\in\{0,1\}^n$ such that $f(g,y)=k$, which tells us the sizes of the blocks are $2^{n-k}\times 2^{n-k}$. Motivated by these observations we define an orthonormal set of vectors in $\mathbb{R}^{2^n}$ by $v_{n+1}:=e_g$, and for all $k\in[n]$ we let $v_k:=\sum_{y:f(g,y)=k}\frac{e_y}{\sqrt{2^{n-k}}}$.

Since the row and column spaces of Γ_g^U are spanned by $\{v_k : k\in[n+1]\}$, we can project on the span of the v_k without changing Γ_g^U. This allows us to reduce Γ_g^U to an $(n+1)\times(n+1)$-dimensional matrix G:

$$\begin{aligned}\Gamma_g^U &= \left(\sum_{k=1}^{n+1} v_k v_k^T\right)\Gamma_g^U\left(\sum_{\ell=1}^{n+1} v_\ell v_\ell^T\right)\\ &= \left(\sum_{k=1}^{n+1} v_k e_k^T\right)\underbrace{\left(\sum_{k=1}^{n+1} e_k v_k^T\right)\Gamma_g^U\left(\sum_{\ell=1}^{n+1} v_\ell e_\ell^T\right)}_{G:=}\left(\sum_{\ell=1}^{n+1} e_\ell v_\ell^T\right).\end{aligned}$$

It follows from the above identity, together with the orthonormality of the vectors $\{v_1,\dots,v_n,v_{n+1}\}$, that

$$\left\|\Gamma_g^U\right\| = \left\|\left(\sum_{k=1}^{n+1} e_k v_k^T\right)\Gamma_g^U\left(\sum_{\ell=1}^{n+1} v_\ell e_\ell^T\right)\right\| = \|G\|. \tag{7.4}$$

Note that $G\in\mathbb{R}^{(n+1)\times(n+1)}$ is a strictly upper-triangular matrix, with the following entries for $k,\ell\in[n]$:

$$\begin{aligned}G_{k,\ell} &= v_k^T\Gamma_g^U v_\ell\\ &= \left(\sum_{z:f(g,z)=k}\frac{e_z^T}{\sqrt{2^{n-k}}}\right)\Gamma_g^U\left(\sum_{z':f(g,z')=\ell}\frac{e_{z'}}{\sqrt{2^{n-\ell}}}\right)\\ &= \frac{2^{\frac{k+\ell}{2}}}{2^n}\left(\sum_{z:f(g,z)=k} e_z^T\right)\Gamma_g^U\left(\sum_{z':f(g,z')=\ell} e_{z'}\right)\\ &= \frac{2^{\frac{k+\ell}{2}}}{2^n}\sum_{z:f(g,z)=k}\ \sum_{z':f(g,z')=\ell}\left(\Gamma_g^U\right)_{z,z'}\\ &= \frac{2^{\frac{k+\ell}{2}}}{2^n}\sum_{z:f(g,z)=k}\ \sum_{z':f(g,z')=\ell} 2^k[k<\ell] && \text{(by (7.2))}\\ &= \frac{2^{\frac{k+\ell}{2}}}{2^n}2^{n-k}2^{n-\ell}2^k[k<\ell]\\ &= 2^{n-\frac{\ell-k}{2}}[k<\ell].\end{aligned}$$

Similarly for $\ell = n+1$ we get that $G_{k,\ell} = \sqrt{2}2^{n-\frac{\ell-k}{2}}[k<\ell]$ for all $k \in [n+1]$. Note that the matrix entries only depend on the difference $\ell - k$. For each $d \in [n]$ define $G_d \in \mathbb{R}^{(n+1)\times(n+1)}$ such that $(G_d)_{k,\ell} = G_{k,\ell}[d = \ell - k]$. This G_d is only non-zero on one non-main diagonal (namely the (k,ℓ)-entries where $d = \ell - k$), and its non-zero entries are all upper bounded by $\sqrt{2}2^n 2^{-\frac{d}{2}}$. We have $G = \sum_{d=1}^n G_d$ and therefore

$$\|G\| \le \sum_{d=1}^{n} \|G_d\| \le \sum_{d=1}^{n} \sqrt{2}2^n 2^{-\frac{d}{2}} = 2^n \sum_{d=0}^{n-1} \left(\sqrt{2}\right)^{-d} \le \frac{2^n}{1-1/\sqrt{2}} \le 2^{n+2}. \tag{7.5}$$

Inequalities (7.3)-(7.5) give that $\left\|\Gamma_g\right\| \le 2^{n+3}$ and hence Lemma 6.5 yields a lower bound of $\Omega\left(\frac{n2^n}{2^{n+3}}\right) = \Omega(n)$ on the number of quantum queries to O_z needed to learn z. □

Theorem 7.4 *Finding a point in $B_\infty(K, 1/7)$ for an unknown convex set K such that $K \subseteq B_\infty(0,2) \subseteq \mathbb{R}^n$ requires $\Omega(n)$ quantum queries to a separation oracle $\mathrm{SEP}_{0,0}(K)$, even if we are promised there exists some unknown $x \in \mathbb{R}^n$ such that $B_\infty(x, 1/3) \subseteq K$.*

Proof. We will prove an $\Omega(n)$ quantum query lower bound for this problem by a reduction from learning with first-difference queries. Let $z \in \{0,1\}^n$ be an unknown binary string, and let us define $K_z := B_\infty(z, 1/3) \subset \mathbb{R}^n$ as a small box around the corner of the hypercube corresponding to z. Then clearly $K_z \subset B_\infty(0,2)$, and finding a point close enough to K_z is enough to recover z.

We can easily reduce a separation oracle query to a first-difference query to z, as follows. Suppose y is the vector that we need to answer a SEP query for:

1. If $y \notin [-1/3, 4/3]^n$, then output a hyperplane separating y from $[-1/3, 4/3]^n$.
2. If $y \in [-1/3, 4/3]^n$, then let g be the nearest corner of the hypercube.
3. Let i be the result of a first-difference query to z with g.
   (a) If the query returns $n+1$, indicating that $z = g$, then we know K_z exactly, so we can find a separating hyperplane or conclude that $y \in K_z$.
   (b) If $z \neq g$, then return e_i if $g_i = 1$, and $-e_i$ if $g_i = 0$.

Hence our $\Omega(n)$ quantum lower bound on learning z with first-difference queries implies an $\Omega(n)$ lower bound on the number of quantum queries to a separation oracle needed for finding a point close to a convex set. □

Since optimization over a set K gives a point close to the set K, this also implies a lower bound on the number of separation queries needed for optimization. This theorem is tight up to logarithmic factors, since it is known that $\widetilde{\mathcal{O}}_{\frac{R}{r\varepsilon}}(n)$ classical separation queries suffice for optimization, even without knowing a point in the convex set [LSW15]. Finally we remark that, due to our improved algorithm for optimization using validity queries that follows from Section 3.5, this also gives an $\widetilde{\Omega}(n)$ lower bound on the number of separation queries needed to implement validity.[4]

[4] It is easy to modify Theorem 7.3 to prove a lower bound on computing the majority function of z,

7.5 | An open question

The main open question remaining is whether we can improve our $\Omega(\sqrt{n})$ lower bound on the number of separation queries needed to implement an optimization oracle when an interior point of K is known. We conjecture that the correct bound is $\tilde{\Theta}(n)$, in which case knowing a point in K would not yield a benefit for query complexity.

We propose *linear system solving* as a candidate for proving an $\Omega(n)$ quantum query lower bound. In particular, for an $n \times n$ matrix A, we conjecture that $\Omega(n)$ quantum queries of the form $|x\rangle|0\rangle \mapsto |x\rangle|Ax\rangle$ are needed to find an x such that $Ax = b$.[5] In Chapter 8 we will show that the corresponding lower bound for finite fields holds, although this has no direct implications for convex optimization. For the real-valued problem no non-trivial lower bound is known, let alone an $\Omega(n)$ lower bound. It might even be that there is an exponential speedup for this problem, which could have many applications in convex optimization, machine learning and many other fields.

which would imply an $\Omega(n)$ lower bound on the number of separation queries needed to implement a validity oracle, without the log factors.

[5] Here the states $|x\rangle$ and $|Ax\rangle$ are binary finite-precision representations of the vectors, *not* an encoding in the amplitudes of the states.

Chapter 8

A quantum query lower bound on the linear Simon's problem

Simon's problem asks the following: determine if a function $f : \{0,1\}^n \to \{0,1\}^n$ is one-to-one or if there exists a unique non-zero $s \in \{0,1\}^n$ such that $f(x) = f(x \oplus s)$ for all $x \in \{0,1\}^n$, given the promise that exactly one of the two holds. Simon [Sim97] showed that a classical bounded-error algorithm that can solve this promise problem for every f requires $\Omega\left(\sqrt{2^n}\right)$ queries to f. In the same paper Simon showed that there is a quantum algorithm that can solve this promise problem using only $\mathcal{O}(n)$ quantum queries to f. A matching lower bound on the number of quantum queries was given in [KNP07], even for a generalization of the problem to functions $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$. We give two short proofs that $\mathcal{O}(n)$ quantum queries is optimal even when we are additionally promised that f is linear. This is somewhat surprising because for linear functions there even exists a *classical* n-query algorithm.

This chapter is mostly based on the paper *Simon's problem for linear functions* by J. van Apeldoorn and S. Gribling [AG18].

8.1 | Introduction

In 1994, Simon [Sim97] showed the existence of a query problem where quantum algorithms offer an exponential improvement over the best randomized bounded-error classical algorithms. The problem he considers is the following:

> *Given a function $f : \{0,1\}^n \to \{0,1\}^n$ with the promise that it either (1) is one-to-one or (2) admits a unique non-zero $s \in \{0,1\}^n$ such that $f(x) = f(x \oplus s)$ for all $x \in \{0,1\}^n$, decide which of the two holds.*

Simon showed that there is a quantum algorithm which can solve this problem for any f satisfying the promise using $\mathcal{O}(n)$ quantum queries to f, i.e., using $\mathcal{O}(n)$ applications of the unitary $|x\rangle|b\rangle \mapsto |x\rangle\big|b \oplus f(x)\big\rangle$.[1] This offers an exponential improvement over classical algorithms, since Simon also showed that at least $\Omega\big(\sqrt{2^n}\big)$ classical queries of the form $x \mapsto f(x)$ are needed in order to succeed with probability at least $2/3$. The question we are interested in is the optimality of Simon's quantum algorithm and its generalization to finite fields. Let p be a prime power and let $\mathbb{F}_p$ be the finite field with p elements. Simon's problem over $\mathbb{F}_p$ can be formulated as follows:

> *Given a function $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$ with the promise that it either (1) is one-to-one or (2) admits a one-dimensional subspace $H \subset \mathbb{F}_p^n$ such that for all $x, y \in \mathbb{F}_p^n$, $f(x) = f(y) \Leftrightarrow x - y \in H$, decide which of the two holds.*

Koiran et al. [KNP07] (for an earlier version see [KNP05]) showed that the quantum query complexity of Simon's problem over $\mathbb{F}_p$ is $\Theta(n)$.[2] Here we show that the lower bound of $\Omega(n)$ quantum queries holds even when f is additionally promised to be *linear*. That is, a quantum algorithm which can solve Simon's problem over $\mathbb{F}_p$ for any linear function requires $\Omega(n)$ quantum queries to f. Interestingly, this shows that for the class of linear functions there is no quantum advantage: classically, one can also fully determine a linear function using n queries. To see this, note that a linear function $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$ corresponds to a matrix $A \in \mathbb{F}_p^{n\times n}$, and a function evaluation is a matrix-vector product. By querying all standard basis vectors we can recover all columns of A and hence know f fully.

Definition 8.1 : Linear Simon's problem *Let p be a prime power. Given a linear function $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$, with the promise on the kernel of f that either $|\ker\big(f\big)| = 1$ or $|\ker\big(f\big)| = p$, decide which of the two holds.*

Our main result (proved in Section 8.2) is the following.

[1] In fact, Simon considered the problem of finding the non-zero string s, if it exists. Here we focus on the decision version of his problem. However, all upper bounds mentioned are derived from algorithms which also find s.

[2] They even prove the analogous lower bound for the hidden subgroup problem over Abelian groups, see our Section 8.4.

Theorem 8.2 *Let p be a prime power. Let $\mathscr{A}$ be a T-query bounded-error quantum algorithm for the linear Simon's problem over $\mathbb{F}_p$. Then $T = \Omega(n)$.*

In Theorem 8.2 we follow the same proof structure as [KNP07], using the polynomial method (see Section 6.3). More specifically, we show that, averaged over a subset of functions, the acceptance probability of a T-query quantum algorithm is a polynomial of degree at most $2T$ in the size of the kernel. We then obtain the lower bound by appealing to [KNP07, Lem. 5] which states that any polynomial with the correct success probabilities has degree $\Omega(n)$. However, where [KNP07] average over all functions, we only consider linear functions over $\mathbb{F}_p^n$. Surprisingly, this simplifies the proof substantially. We also give a slightly simplified proof of [KNP07, Lem. 5].

In Section 8.3 we will give an alternative and shorter method for proving Theorem 8.2 when p is a prime (note that in the rest of this chapter p may be a *prime power*). This shorter method uses a known lower bound on the communication complexity of the linear Simon's problem. The alternative proof is not included in [AG18] since we only recently found it.

Notation For the rest of this chapter we define $F := \{f : \mathbb{F}_p^n \to \mathbb{F}_p^n \mid f \text{ linear}\}$ as the set of all linear functions from $\mathbb{F}_p^n$ to $\mathbb{F}_p^n$. For each $k \in \{0, 1, \ldots, n\}$ and $D = p^k$ we let F_D be the subset of F consisting of linear functions whose kernel has size D, i.e., $F_D = \{f \in F \mid |\ker(f)| = D\}$.

8.2 | Proof of Theorem 8.2

Let $\mathscr{A}$ be a T-query bounded-error algorithm for the linear Simon's problem and let $P(f)$ be the acceptance probability of $\mathscr{A}$ on the input f. As in Section 6.3, we can write $P(f)$ as a polynomial in the $\delta_{f(x),y}$:

$$P(f) = \sum_{\substack{s \subseteq \mathbb{F}_p^n \times \mathbb{F}_p^n \\ |s| \le 2T}} \beta_s \prod_{(x,y) \in s} \delta_{f(x),y}.$$

When we view s as a partial function, this expression can be rewritten in terms of f extending s: (remember that we write $s \preceq f$ if f extends s)

$$P(f) = \sum_{s \in S_{2T}} \beta_s [s \preceq f],$$

where S_{2T} is the set of all partial functions s with $|\mathrm{dom}(s)| \le 2T$. As in the example in Section 6.3, it will be useful to average $P(f)$ over certain sets of inputs. We consider the average acceptance probability $Q(D)$ over all functions with a kernel of size D:

$$\begin{aligned}Q(D) &= \sum_{f\in F_D}\frac{1}{|F_D|}P(f)\\ &= \sum_{f\in F_D}\frac{1}{|F_D|}\sum_{s\in S}\beta_s[s\preceq f]\\ &= \sum_{s\in S}\beta_s\frac{1}{|F_D|}\sum_{f\in F_D}[s\preceq f]\\ &= \sum_{s\in S}\beta_s Q_s(D).\end{aligned}$$

Here $Q_s(D)$ is the probability that a uniformly random $f\in F_D$ extends s:

$$Q_s(D) = \frac{1}{|F_D|}\sum_{f\in F_D}[s\preceq f] = \Pr_{f\in_R F_D}[s\preceq f].$$

In the next two sections we will prove that the degree of Q needs to be at least linear in n, and that the degree of each Q_s (and hence of Q) is upper bounded by $2T$. Together these results imply Theorem 8.2.

8.2.1 | Lower bound on the degree

For $k\in\{0,1,\dots,n\}$, $Q(p^k)$ represents an acceptance probability and therefore $Q(p^k)\in[0,1]$. Moreover, if the algorithm succeeds with probability at least $2/3$, then $Q(1)\geq 2/3$ and $Q(p)\leq 1/3$. The lemma below was already present in [KNP07] and shows that such a Q has degree $\Omega(n)$. We give a slightly simplified proof for completeness.

Lemma 8.3 : [KNP07, Lem. 5] *For every univariate polynomial Q such that $Q(1)\geq 2/3$, $Q(p)\leq 1/3$ and $Q(p^k)\in[0,1]$ for all $k\in\{0,\dots,n\}$, it holds that $\deg(Q)\geq n/4$.*

Proof. We start by noticing that by the mean value theorem we know that there is an $x_0\in[1,p]$ for which $|Q'(x_0)|\geq\frac{1}{3(p-1)}$. Assume that Q is a polynomial of degree $d\leq n/2$ (otherwise we are done), so that its derivative Q' is of degree $d-1$ and its second derivative Q'' is of degree $d-2$. Consider the $2d-2$ intervals of the form (p^a,p^{a+1}) where $a=n-(2d-2),\dots,n-1$. Since together Q' and Q'' have at most $2d-3$ roots, there is such an interval for which both polynomials have no roots with real part in it; let $a\geq n-(2d-2)$ be the integer corresponding to this interval and let $M:=\frac{1+p}{2}p^a$ be the middle of this interval. To show the degree lower bound it suffices to prove the following chain of inequalities:

$$\frac{1}{p^{2d-2}}\overset{(*)}{\leq}\left|\frac{Q'(M)}{Q'(x_0)}\right|\overset{(**)}{\leq}\frac{6}{p^{n-2d+2}}.$$

Indeed, if the above chain of inequalities holds, then $6\geq p^{n-4d+4}\geq 2^{n-4d+4}$ which implies that $n-4d+4\leq 3$, i.e., $d\geq\frac{n+1}{4}$. We now prove $(*)$ and $(**)$ separately.

$(*)$ For the lower bound we will use the following elementary fact:

$$\text{if } 0 \le v < w \text{ and } 0 \le y, \text{ then } \frac{v+y}{w+y} \ge \frac{v}{w}. \tag{8.1}$$

Denote the roots of Q' by $b_j + c_j\mathbf{i}$, for $j \in [d-1]$. Then $Q'(x) = \lambda \prod_{j=1}^{d-1}(x - b_j - c_j\mathbf{i})$ for some $\lambda \in \mathbb{R}$ and hence

$$\left|\frac{Q'(M)}{Q'(x_0)}\right| = \left|\prod_{j=1}^{d-1} \frac{M - b_j - c_j\mathbf{i}}{x_0 - b_j - c_j\mathbf{i}}\right| = \prod_{j=1}^{d-1}\left|\frac{M - b_j - c_j\mathbf{i}}{x_0 - b_j - c_j\mathbf{i}}\right| = \prod_{j=1}^{d-1}\sqrt{\frac{(M-b_j)^2 + c_j^2}{(x_0-b_j)^2 + c_j^2}}.$$

We will show that each factor in the product is bounded from below by $1/p^2$. Considering the j-th factor. If $|x_0 - b_j| \le |M - b_j|$, then we are clearly done. Hence, assume $|x_0 - b_j| > |M - b_j|$. This implies that

$$\begin{aligned} b_j &> x_0 + |M - b_j| \\ &\ge x_0 + \frac{|M - x_0|}{2} \\ &> \frac{|M - x_0|}{2} \\ &\ge p^{a-1}. \end{aligned}$$

We now use (8.1):

$$\sqrt{\frac{(M-b_j)^2 + c_j^2}{(x_0-b_j)^2 + c_j^2}} \ge \left|\frac{M - b_j}{x_0 - b_j}\right|.$$

Since we know that $b_j > p^{a-1}$, and because by the definition of a there is no $b_j \not\in (p^a, p^{a+1})$, there are two cases to consider:

- If $b_j \in (p^{a-1}, p^a]$, then

$$\left|\frac{M - b_j}{x_0 - b_j}\right| \ge \inf_{x \in (p^{a-1}, p^a)} \left|\frac{M - x}{x_0 - x}\right| = \left|\frac{M - p^a}{x_0 - p^a}\right| \ge \frac{1}{2} \ge \frac{1}{p^2}.$$

- If $b_j \in [p^{a+1}, \infty)$, then

$$\left|\frac{M - b_j}{x_0 - b_j}\right| = \frac{-\frac{1+p}{2}p^a + b_j}{-x_0 + b_j} = \frac{\frac{p-1}{2}p^a + (b_j - p^{a+1})}{p^{a+1} - x_0 + (b_j - p^{a+1})} \ge \frac{p^{a-1}}{p^{a+1} - x_0} \ge \frac{1}{p^2},$$

  where we use (8.1) and $\frac{p-1}{2} \ge \frac{1}{p}$ for the first inequality.

$(**)$ By construction we have $|Q'(x_0)| \geq \frac{1}{3(p-1)}$, so it remains to show that $|Q'(M)| \leq (\frac{p-1}{2}p^{n-2d+2})^{-1}$. Assume towards a contradiction that $|Q'(M)| > (\frac{p-1}{2}p^{a})^{-1}$. Since Q'' has no roots with real part in the interval (p^a, p^{a+1}), Q' is either strictly increasing or strictly decreasing on the interval (p^a, p^{a+1}). Therefore, there is an interval (α, β) (with $(\alpha, \beta) = (p^a, M)$ or $(\alpha, \beta) = (M, p^{a+1})$) of length $\frac{p-1}{2}p^a$ where $|Q'(x)| > \left(\frac{p-1}{2}p^a\right)^{-1}$. By the fundamental theorem of calculus this implies that $|Q(\alpha) - Q(\beta)| > 1$. Since $Q'(x)$ does not have any roots with real value in (p^a, p^{a+1}) we know that $Q(x)$ is monotone on this interval. We get a contradiction, since we have $1 \geq |Q(p^{a+1}) - Q(p^a)| \geq |Q(\alpha) - Q(\beta)|$. It follows that

$$|Q'(M)| \leq \left(\frac{p-1}{2}p^a\right)^{-1} \leq \left(\frac{p-1}{2}p^{n-2d+2}\right)^{-1}.$$

We conclude that $\frac{1}{p^{2d-2}} \leq \left|\frac{Q'(M)}{Q'(x_0)}\right| \leq \frac{3(p-1)}{\frac{p-1}{2}p^{n-2d+2}}$ and hence that $d \geq n/4$. □

8.2.2 | Upper bound on the degree

We now show that the degree of each Q_s is upper bounded by $2T$.

Lemma 8.4 *Let $s : \mathrm{dom}(s) \subseteq \mathbb{F}_p^n \to \mathbb{F}_p^n$ be a partial linear function, then*

$$\deg_D(Q_s) \leq \dim\big(\mathrm{span}(\mathrm{dom}(s))\big)$$

Proof. Let $K := \mathrm{span}(\mathrm{dom}(s))$ and $k := \dim(K)$. We can extend s uniquely to a linear function on K. Define $Z := \ker(s) \subseteq K$ and $z := \dim(Z)$, and $Y := Z^\perp \cap K$. For a function $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$ in F_D we write $H := \ker\big(f\big)$, $h := \dim(H)$ and hence $D = |H| = p^h$. We show that $\Pr_{f \in_R F_D}\big[s \preceq f\big]$ has degree at most k as a polynomial in D. If f extends s, then the kernel of s has to be a subset of f. Furthermore, all points where s is defined that are not in the kernel of s cannot be in the kernel of f. Using this observation we analyze the probability $\Pr_{f \in_R F_D}\big[s \preceq f\big]$ in three parts:

$$\begin{aligned}\Pr_{f \in_R F_D}\big[s \preceq f\big] &= \Pr_{f \in_R F_D}[Z \subseteq H \wedge Y \cap H = \{0\}] \\ &\qquad\qquad \cdot \Pr_{f \in_R F_D}\big[s \preceq f \mid Z \subseteq H \wedge Y \cap H = \{0\}\big] \\ &= \Pr_{f \in_R F_D}[Z \subseteq H] \cdot \Pr_{f \in_R F_D}[Y \cap H = \{0\} \mid Z \subseteq H] \\ &\qquad\qquad \cdot \Pr_{f \in_R F_D}\big[s \preceq f \mid Z \subseteq H \wedge Y \cap H = \{0\}\big].\end{aligned}$$

We show that

(1) $\Pr_{f \in_R F_D}[Z \subseteq H]$ is a polynomial in D of degree at most z,

(2) $\Pr_{f \in_R F_D}[Y \cap H = \{0\} \mid Z \subseteq H]$ is a polynomial in D of degree at most $k - z$,

(3) $\Pr_{f \in_R F_D}\big[s \preceq f \mid Z \subseteq H \wedge Y \cap H = \{0\}\big]$ does not depend on D.

Together, this implies that $\Pr_{f \in_R F_D}\big[s \preceq f\big]$ is a polynomial in D of degree at most k.

(1) $\Pr_{f\in_R F_D}[Z\subseteq H]$ equals the fraction of h-dimensional subspaces of $\mathbb{F}_p^n$ that contain Z. There are $\alpha(n,h) := \frac{1}{h!}\prod_{i=0}^{h-1}(p^n - p^i)$ ways to pick h linearly independent vectors in a space of dimension n. Each way gives us a basis for an h-dimensional subspace, however, each h-dimensional subspace has $\alpha(h,h)$ different bases. Hence there are $\frac{\alpha(n,h)}{\alpha(h,h)}$ different subspaces of dimension h in $\mathbb{F}_p^n$. The number of h-dimensional subspaces that contain Z equals the number of $(h-z)$-dimensional subspaces in an $(n-z)$-dimensional space. Hence

$$\Pr_{f\in_R F_D}[Z\subseteq H] = \frac{\frac{\alpha(n-z,h-z)}{\alpha(h-z,h-z)}}{\frac{\alpha(n,h)}{\alpha(h,h)}} = \frac{\prod_{i=0}^{h-z-1}\frac{p^{n-z}-p^i}{p^{h-z}-p^i}}{\prod_{i=0}^{h-1}\frac{p^n-p^i}{p^h-p^i}} = \frac{\prod_{i=z}^{h-1}\frac{p^n-p^i}{p^h-p^i}}{\prod_{i=0}^{h-1}\frac{p^n-p^i}{p^h-p^i}} = \prod_{i=0}^{z-1}\frac{p^h-p^i}{p^n-p^i},$$

which is a degree-z polynomial in terms of $D = p^h$.

(2) We have

$$\Pr_{f\in_R F_D}[Y\cap H = \{0\} \mid Z\subseteq H] = \Pr_{f\in_R F_D}[Y/Z\cap H/Z = \{0\}]$$

where both Y/Z and H/Z are subspaces of $\mathbb{F}_p^n/Z \simeq \mathbb{F}_p^{n-z}$. By construction we have that $\dim(Y/Z) = \dim(Y) = k-z$, $\dim(H/Z) = h-z$. The probability $\Pr_{f\in_R F_D}[Y/Z\cap H/Z = \{0\}]$ equals the number of $(h-z)$-dimensional subspaces of $\mathbb{F}_p^{n-z}$ which are linearly independent from Y/Z, divided by the total number of $(h-z)$-dimensional subspaces. The number of $(h-z)$-dimensional subspaces of $\mathbb{F}_p^{n-z}$ which are linearly independent from Y/Z is equal to the number of ways to pick a basis for such a subspace, which is $\frac{1}{(h-z)!}\left(\prod_{i=0}^{h-z-1} p^{n-z} - p^{k-z+i}\right)$, divided by the number of such bases, which is $\alpha(h-z,h-z)$. We get

$$\begin{aligned}
\Pr_{f\in_R F_D}[Y/Z\cap H/Z = \{0\}] &= \frac{\frac{\frac{1}{(h-z)!}\prod_{i=0}^{h-z-1}(p^{n-z}-p^{k-z+i})}{\alpha(h-z,h-z)}}{\frac{\alpha(n-z,h-z)}{\alpha(h-z,h-z)}} \\
&= \frac{\frac{1}{(h-z)!}\prod_{i=0}^{h-z-1}\left(p^{n-z}-p^{k-z+i}\right)}{\alpha(n-z,h-z)} \\
&= \frac{\prod_{i=0}^{h-z-1}\left(p^{n-z}-p^{k-z+i}\right)}{\prod_{i=0}^{h-z-1}\left(p^{n-z}-p^{i}\right)} \\
&= \frac{\prod_{i=k-z}^{h+k-2z-1}\left(p^{n-z}-p^{i}\right)}{\prod_{i=0}^{h-z-1}\left(p^{n-z}-p^{i}\right)} \\
&\overset{(a)}{=} \frac{\prod_{i=h-z}^{h+k-2z-1}\left(p^{n-z}-p^{i}\right)}{\prod_{i=0}^{k-z-1}\left(p^{n-z}-p^{i}\right)} \\
&= \frac{\prod_{i=0}^{k-z-1}\left(p^{n-z}-p^{i+h-z}\right)}{\prod_{i=0}^{k-z-1}\left(p^{n-z}-p^{i}\right)}.
\end{aligned}$$

To see equality (a), if $k < h$ then cancel the terms for $i = k - z, \dots, h - z$ on both sides of the fraction, and if $k > h$ then add the terms for $i = h - z, \dots, k - z$ on both sides of the fraction.

It follows that

$$\Pr_{f \in_R F_D}[Y/Z \cap H/Z = \{0\}] = \frac{\prod_{i=0}^{k-z-1}\left(p^{n-z} - p^h p^{-z-i}\right)}{\prod_{i=0}^{k-z-1}\left(p^{n-z} - p^i\right)}$$

is a polynomial in $D = p^h$ of degree $k - z$. We mention in passing that, alternatively, one can arrive at the same expression by looking at the probability that a random Y is linearly independent from a fixed H.

(3) Finally we consider $\Pr_{f \in_R F_D}\left[s \preceq f \mid Z \subseteq H \wedge Y \cap H = \{0\}\right]$. Since $Z \subseteq H$, we know that f and s agree on Z. Hence, f extends s if their values agree on Y. Let $b_1, \dots, b_{k-z}$ be a basis for Y, then f and s agree on Y if and only if they agree on $b_1, \dots, b_{k-z}$. Since we condition on the event $Y \cap H = \{0\}$, the probability that this happens does not depend on $D = p^h$.

□

8.3 | An alternative proof of Theorem 8.2

An alternative proof of Theorem 8.2 can be given via lower bound results in quantum communication complexity[3]. This approach was hinted at in [SWYZ19] as a method for lower bounding the number of matrix-vector queries needed to determine certain properties of a matrix. It should be noted that this proof only works if p is a prime, while the other proof also works if p is a prime power. Consider the following problem:

Definition 8.5 : Linear Simon's communication problem *Let p be a prime, $A, B \in \mathbb{F}_p^{n\times n}$ and $M = A + B$ be such that* $\mathrm{rank}(M) \in \{n-1, n\}$. *Let A be given to Alice and B to Bob. The* linear Simon's communication problem *is the communication problem of deciding the rank of M.*

In [LSWW14] this problem is introduced as $\mathrm{RANK}_{n,n-1}$ and a lower bound is proven on the randomized communication complexity. However, as noted by the authors, the proof given in [LSWW14] actually lower bounds the quantum communication complexity as well, leading to the following lemma:

Lemma 8.6 : [LSWW14, Thm. 1] *Let p be a prime. Every protocol that solves the linear Simon's communication problem with success probability at least* $9/10$ *requires* $\Omega(n^2\log(p))$ *qubits of quantum communication.*

We are now ready to state the alternative proof of Theorem 8.2.

[3] For an introduction into quantum communication complexity, see for example [Wol02].

Alternative proof of Theorem 8.2 when p is a prime. Let p be a prime and $\mathscr{A}$ be a T-query quantum algorithm for the linear Simon's problem over $\mathbb{F}_p^n$. Consider an instance (A,B) of the linear Simon's communication problem. To solve the communication problem Alice executes the algorithm $\mathscr{A}$ for $M = A+B$. To implement a query Alice and Bob use $\mathscr{O}\big(n\log(p)\big)$ qubits of communication: Alice sends $|x\rangle|0\rangle$ and bob replies with $|x\rangle|Bx\rangle$, which allows Alice to compute $|(A+B)x\rangle$, after which they uncompute $|Bx\rangle$. In total the protocol will use $\mathscr{O}\big(Tn\log(p)\big)$ qubits of communication to solve the linear Simon's problem for M, and hence to solve the linear Simon's communication problem for (A,B). We conclude that $T = \Omega(n)$. □

The restriction that p is a prime in Lemma 8.6 might be liftable by a careful analysis of the proof in [LSWW14]. This alternative approach has the benefit that a similar communication lower bound over the integers or the real numbers would directly translate to a query lower bound those as well.

8.4 | Open problems

To conclude, we mention the following open problems:

- Koiran et al. [KNP07] lift the lower bound on Simon's problem over $\mathbb{F}_p^n$ to the hidden subgroup problem over finite Abelian groups:

  > *Given a (finite Abelian) group G and a function $f : G \to X$ with the promise that there is a subgroup $H \leq G$ of rank either 0 or 1 (i.e., either trivial, or generated by a single element), such that $f(g) = f(g')$ if and only if $g - g' \in H$, decide which of the two holds.*

  One recovers Simon's problem over $\mathbb{F}_p^n$ by taking $G = X = \mathbb{F}_p^n$. A natural question is whether the hidden subgroup problem over finite Abelian groups also remains equally hard when we are additionally promised that f is an endomorphism. The reduction used by Koiran et al. combined with our result gives a smaller and more structured set of hard instances of the hidden subgroup problem over Abelian groups. However, the functions obtained from this reduction will only be endomorphisms on a subgroup of G, not on all of G.

- While the general Simon's problem has no natural extension to $\mathbb{R}^n$, the linear Simon's problem can possibly be extended to $\mathbb{R}^n$. For example: given matrix-vector multiplication queries $x \mapsto Ax$ for a psd matrix A with $\|A\| \leq 1$, and given an error parameter $\varepsilon > 0$, decide if $\lambda_{\min}(A) \leq \varepsilon$ or $\lambda_{\min}(A) \geq 2\varepsilon$. It remains an open question to prove a lower bound on this problem. As mentioned in Section 7.5, an $\Omega(n)$ lower bound for this problem could resolve whether there is a speedup for implementing an optimization oracle using a separation oracle.

- Aaronson and Ben-David [AB16] introduced the idea of *sculpting* functions. They characterized the total Boolean functions for which there is a promise on the input such that restricted to that promise there is an exponential separation between quantum and classical query complexity. We propose the related idea of *over-sculpting*: bringing the classical query complexity down to the quantum query complexity. More specifically, for which (possibly partial) Boolean functions f does there exist a promise P such that:

  $$Q_{1/3}(f) \leq o\big(R_{1/3}(f)\big)$$

  $$Q_{1/3}(f) = \Theta\big(Q_{1/3}(f|_P)\big) = \Theta\big(R_{1/3}(f|_P)\big).$$

  Simon's problem does not correspond to a Boolean function since the input alphabet is not Boolean[4], but our results show that Simon's problem can be oversculpted in this slightly different setting.

- Although we know that over $\mathbb{F}_p$ we need to query a matrix fully in order to determine its rank, the time complexity of this problem is not yet known, both in the classical and the quantum setting.

[4] An input for Simon's problem is a function $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$, which can be viewed as a string of length p^n over the alphabet $\mathbb{F}_p^n$.

Chapter 9

Limitations of quantum LP and SDP-solvers

We start this chapter by considering the limitations of specific types of quantum LP and SDP-solvers. In Chapter 4 we developed algorithms that had a polynomial dependence on the scale invariant error parameter γ in their complexity. Here we show that for many natural families of LPs and SDPs this γ parameter grows linearly with n and m, making methods with a large dependence on γ unsuitable for solving these problems. We then consider methods that always return sparse solutions. We show that these solutions cannot be useful for solving problems with a lot of symmetry. For our LP and SDP-solvers this implies that a high iteration count (and hence a large γ parameter) is needed in order to find a useful solution for certain problems.

In the second part of this chapter we give six different quantum query lower bounds for LP and SDP-solving. These bounds show that in the sparse matrix input model for SDPs our n, m and s-dependence is optimal, and that a polynomial dependence on γ is required in order to achieve an $o(nm)$ dependence on n and m. We also give lower bounds on the query complexity of solving non-negative LPs, a special subclass of LPs. Here we show that for certain non-negative LPs the classical algorithms are already optimal in the quantum setting. We finish the chapter by showing quantum query lower bounds for the Hamiltonian input model, the quantum operator input model, and the quantum state input model.

This chapter is mostly based on the papers “Quantum SDP-Solvers: Better upper and lower bounds” by J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf [AGGW17] and “Improvements in Quantum SDP-Solving with Applications” by J. van Apeldoorn and A. Gilyén [AG19a], as well as on some unpublished results.

9.1 | Downsides of specific methods

In this section we show some of the limitations of our methods and methods like them. We start by showing that many natural families of SDPs have a γ parameter that increases linearly with n and m. This implies that quantum SDP-solvers with a large polynomial dependence on γ, like our SDP-solvers, will be ill-suited for these SDPs.

We then consider SDP-solvers (and LP-solvers) that return sparse dual solutions, like our methods do. We will show, that for problems with a lot of symmetry, a dual solution cannot be too sparse if it contains useful information. Hence, any solver that returns a dual solution for these problems, and that only generates a constant number of non-zero entries of this dual solution per iteration, requires many iterations.

9.1.1 | Families of SDPs with a large γ parameter

In this section we use r^* to denote the smallest ℓ_1-norm of an optimal solution to the dual of an SDP (remember that r only denotes an upper bound), and R^* to denote the smallest trace of an optimal solution to the primal. It turns out that for many natural classes of SDPs the parameters r^*, R^*, ε, n and m can grow linearly for some instances. In particular, this is the case if SDPs in a class combine in a natural manner. Take for example two SDP relaxations for the MAXCUT problem on two graphs $G^{(1)}$ and $G^{(2)}$ (on $n^{(1)}$ and $n^{(2)}$ vertices respectively):

$$\begin{array}{rl} \max & \mathrm{Tr}\big(L(G^{(1)})X^{(1)}\big) \\ \text{s.t.} & \mathrm{Tr}\big(X^{(1)}\big) \leq n^{(1)} \\ & \mathrm{Tr}\big(E_{jj}X^{(1)}\big) \leq 1 \text{ for } j = 1,\ldots,n^{(1)} \\ & X^{(1)} \succeq 0 \end{array} \qquad \begin{array}{rl} \max & \mathrm{Tr}\big(L(G^{(2)})X^{(2)}\big) \\ \text{s.t.} & \mathrm{Tr}\big(X^{(2)}\big) \leq n^{(2)} \\ & \mathrm{Tr}\big(E_{jj}X^{(2)}\big) \leq 1 \text{ for } j = 1,\ldots,n^{(2)} \\ & X^{(2)} \succeq 0. \end{array}$$

Where $L(G)$ is the Laplacian of a graph G, i.e., $L(G) = D(G) - A(G)$ with $A(G)$ the adjacency matrix of G and $D(G)$ the diagonal matrix with the degrees of the vertices on the diagonal. Note that this is not normalized to operator norm ≤ 1, but for simplicity we ignore this here. If we denote the direct sum of two matrices by $\oplus$, that is

$$A \oplus B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix},$$

then, for the disjoint union of the two graphs, we have

$$L(G^{(1)} \cup G^{(2)}) = L(G^{(1)}) \oplus L(G^{(2)}).$$

This, plus the fact that the trace distributes over direct sums of matrices, means that the SDP relaxation for MAXCUT on $G^{(1)} \cup G^{(2)}$ is the same as a natural combination of the two separate maximizations:

$$\begin{aligned}
\max \quad & \operatorname{Tr}\big(L(G^{(1)})X^{(1)}\big) + \operatorname{Tr}\big(L(G^{(2)})X^{(2)}\big) \\
\text{s.t.} \quad & \operatorname{Tr}\big(X^{(1)}\big) + \operatorname{Tr}\big(X^{(2)}\big) \leq n^{(1)} + n^{(2)} \\
& \operatorname{Tr}\big(E_{jj}X^{(1)}\big) \leq 1 \text{ for } j = 1, \ldots, n^{(1)} \\
& \operatorname{Tr}\big(E_{jj}X^{(2)}\big) \leq 1 \text{ for } j = 1, \ldots, n^{(2)} \\
& X^{(1)}, X^{(2)} \succeq 0.
\end{aligned}$$

It is easy to see that the new value of n is $n^{(1)} + n^{(2)}$, the new value of m is $m^{(1)} + m^{(2)} - 1$ and the new value of R^* is $n^{(1)} + n^{(2)} = R^{*(1)} + R^{*(2)}$. Since it is a natural choice for the MAXCUT relaxation to pick the allowed additive error so that they add together under such combinations, all these parameters would increase linearly if we would keep adding instances together. It remains to see what happens to r^*. As we will see later in this section, under some mild conditions, these kinds of combinations imply that there are MAXCUT-relaxation SDPs for which r^* also increases linearly, but this requires a bit more work.

Definition 9.1 : Combineable class of SDPs *We say that a class of SDPs (each with an associated allowed approximation error) is* combinable *with k global constraints if for every two elements in this class, $(SDP^{(a)}, \varepsilon^{(a)})$ and $(SDP^{(b)}, \varepsilon^{(b)})$, there is an instance in the class, $(SDP^{(c)}, \varepsilon^{(c)})$, that is a combination of the two in the following sense:*

- $C^{(c)} = C^{(a)} \oplus C^{(b)}$.
- $A_j^{(c)} = A_j^{(a)} \oplus A_j^{(b)}$ *and* $b_j^{(c)} = b_j^{(a)} + b_j^{(b)}$ *for all* $j \in [k]$.
- $A_j^{(c)} = A_j^{(a)} \oplus \mathbf{0}$ *and* $b_j^{(c)} = b_j^{(a)}$ *for* $j = k+1, \ldots, m^{(a)}$.
- $A_{m^{(a)}+j-k}^{(c)} = \mathbf{0} \oplus A_j^{(b)}$ *and* $b_{m^{(a)}+j-k}^{(c)} = b_j^{(b)}$ *for* $j = k+1, \ldots, m^{(b)}$.
- $\varepsilon^{(c)} \leq \varepsilon^{(a)} + \varepsilon^{(b)}$.

In other words, some fixed set of constraints are summed pairwise, and the remaining constraints get included in the SDP separately.

Note that this is a natural generalization of the combining property of the MAXCUT relaxations (in that case $k = 1$ to account for the trace bound).

Theorem 9.2 *Let k be a non-negative integer. If a class of SDPs is combinable with k global constraints, and there is an element $SDP^{(1)}$ for which every optimal dual solution has the property that*

$$\sum_{j=k+1}^{m^{(1)}} y_j \geq \delta$$

for some $\delta > 0$, then there is a sequence $(SDP^{(t)})_{t\in\mathbb{N}}$ in the class such that $R^{*(t)}$, $r^{*(t)}$, $\varepsilon^{(t)}$, $n^{(t)}$, and $m^{(t)}$ all increase linearly with t. Hence $\gamma^{(t)} = \frac{R^{*(t)} r^{*(t)}}{\varepsilon^{(t)}} = \Omega\big(n^{(t)}\big)$.

Proof. Note that k is fixed and that $k < m^{(1)}$. The sequence we will consider is the t-fold combination of $SDP^{(1)}$ with itself. If for $SDP^{(1)}$ the primal and dual are

$$\begin{array}{rll} \max & \mathrm{Tr}(CX) & \\ \text{s.t.} & \mathrm{Tr}\left(A_j X\right) \leq b_j & \text{for } j \in \left[m^{(1)}\right] \\ & X \succeq 0 & \end{array} \qquad \begin{array}{rl} \min & \sum_{j=1}^{m^{(1)}} b_j y_j \\ \text{s.t.} & \sum_{j=1}^{m^{(1)}} y_j A_j - C \succeq 0 \\ & y \geq 0, \end{array}$$

then $SDP^{(t)}$ is

$$\begin{array}{rll} \max & \sum_{i=1}^{t} \mathrm{Tr}(CX_i) & \\ \text{s.t.} & \sum_{i=1}^{t} \mathrm{Tr}\left(A_j X_i\right) \leq t b_j & \text{for } j \in [k] \\ & \mathrm{Tr}\left(A_j X_i\right) \leq b_j & \text{for } j = k+1, \ldots, m^{(1)} \text{ and } i = 1, \ldots, t \\ & X_i \succeq 0 & \text{for all } i = 1, \ldots, t \end{array}$$

with dual (where the variables are $y \in \mathbb{R}^k$ and $y^1, \ldots, y^t \in \mathbb{R}^{m^{(1)}-k}$)

$$\begin{array}{rl} \min & \sum_{j=1}^{k} t b_j y_j + \sum_{i=1}^{t} \sum_{j=k+1}^{m^{(1)}} b_j y_j^i \\ \text{s.t.} & \sum_{j=1}^{k} y_j A_j + \sum_{j=k+1}^{m^{(1)}} y_j^i A_j \succeq C \text{ for } i = 1, \ldots, t \\ & y \geq 0 \text{ and } y^i \geq 0 \text{ for all } i \in [t]. \end{array}$$

First, let us consider the value of $\mathrm{OPT}^{(t)}$. Let $X^{(1)}$ be an optimal solution to $SDP^{(1)}$ and for all $i \in [t]$ let $X_i = X^{(1)}$. Since these X_i form a feasible solution to $SDP^{(t)}$, this shows that $\mathrm{OPT}^{(t)} \geq t \cdot \mathrm{OPT}^{(1)}$. Furthermore, let $y^{(1)}$ be an optimal dual solution of $SDP^{(1)}$, then $\left(y_1^{(1)}, \ldots, y_k^{(1)}\right) \oplus \left(y_{k+1}^{(1)}, \cdots, y_{m^{(1)}}^{(1)}\right)^{\oplus t}$ is a feasible dual solution for $SDP^{(t)}$ with objective value $t \cdot \mathrm{OPT}^{(1)}$, so by weak duality $\mathrm{OPT}^{(t)} = t \cdot \mathrm{OPT}^{(1)}$.

Next, let us consider the value of $r^{*(t)}$. Let $\tilde{y} \oplus y^1 \oplus \cdots \oplus y^t$ be an optimal dual solution for $SDP^{(t)}$, split into the parts of y that correspond to different parts of the combination. Then $\tilde{y} \oplus y^i$ is a feasible dual solution for $SDP^{(1)}$ and hence $b^T(\tilde{y} \oplus y^i) \geq \mathrm{OPT}^{(1)}$. On the other hand we have

$$t \cdot \mathrm{OPT}^{(1)} = \mathrm{OPT}^{(t)} = \sum_{i=1}^{t} b^T(\tilde{y} \oplus y^i),$$

this implies that each term in the sum is actually equal to $\mathrm{OPT}^{(1)}$. But if $(\tilde{y} \oplus y^i)$ is an optimal dual solution of $SDP^{(1)}$ then $\left\|(\tilde{y} \oplus y^i)\right\|_1 \geq r^{*(1)}$ by definition, and $\left\|y^i\right\|_1 \geq \delta$. We conclude that $r^{*(t)} \geq r^{*(1)} - \delta + t\delta$.

Now we know the behavior of r^* under combinations, let us look at the primal to find a similar statement for $R^{*(t)}$. Define a new SDP, $\widehat{SDP}^{(t)}$, in which all the constraints are summed when combining, that is, in Definition 9.1 we take $k = m^{(1)}$ and also sum the psd constraints:

$$\begin{aligned} \max \quad & \sum_{i=1}^{t} \operatorname{Tr}(CX_i) \\ \text{s.t.} \quad & \sum_{i=1}^{t} \operatorname{Tr}\left(A_j X_i\right) \le t b_j \quad \text{for } j \in \left[m^{(1)}\right] \\ & \sum_{i=1}^{t} X_i \succeq 0. \end{aligned}$$

This SDP has the same objective function as $SDP^{(t)}$ but a larger feasible region: every feasible $X_1,\ldots,X_t$ for $SDP^{(t)}$ is also feasible for $\widehat{SDP}^{(t)}$. However, by a change of variables, $X := \sum_{i=1}^{t} X_i$, it is easy to see that $\widehat{SDP}^{(t)}$ is simply a scaled version of $SDP^{(1)}$. So, $\widehat{SDP}^{(t)}$ has optimal value $t\cdot\mathrm{OPT}^{(1)}$. Since optimal solutions to $\widehat{SDP}^{(t)}$ are scaled optimal solutions to $SDP^{(1)}$, we have $\hat{R}^{*(t)} = t \cdot R^{*(1)}$. Combining the above, it follows that every optimal solution to $SDP^{(t)}$ is optimal to $\widehat{SDP}^{(t)}$ as well (up to scaling with a factor t), and hence has trace at least $t \cdot R^{*(1)}$, so $R^{*(t)} \ge t \cdot R^{*(1)}$.

We conclude that (since $\varepsilon^{(t)} \le t\varepsilon^{(1)}$ by assumption)

$$\frac{R^{*(t)} r^{*(t)}}{\varepsilon^{(t)}} \ge \frac{tR^{*(1)}(r^{*(1)} + (t-1)\delta)}{t\varepsilon^{(1)}} = \Omega(t)$$

and $n^{(t)} = tn^{(1)}$, $m^{(t)} = t(m^{(1)} - k) + k$. □

This shows that for many natural SDP formulations for combinatorial problems, such as the MAXCUT relaxation, or LPs that have to do with resource management, $\gamma = \Omega(n)$ for some instances. Methods with a large polynomial dependence on γ, like our LP and SDP-solvers, are therefore not well suited for these problems.

9.1.2 | Sparse methods are restrictive

Our LP and SDP-solvers have the property that they return sparse dual solutions, mainly $\widetilde{\mathcal{O}}_n\left(\gamma^2\right)$-sparse dual solutions. In this section we consider to what extent this limits their usefulness. Of course it could be the case that almost every SDP of interest has a sparse approximate dual solution (or can easily be rewritten so that it does), and hence sparseness might be not a restriction at all. However, as we will see below, this is not the case. We will prove that for certain kinds of SDPs, no “useful” dual solution can be very sparse. Intuitively, a dual solution to an SDP is “useful” if it can be turned into a solution of the problem that the SDP is trying to solve. We make this more precise in the definition below. This lower bound on the sparseness of a dual solution for these problems implies a lower bound on γ. In cases where this lower bound is large, our LP and SDP-solvers will have a bad bound on the complexity.

Definition 9.3 : Solving a problem via the dual *A problem is defined by a function f that, for every element p of the problem domain $\mathscr{D}$, gives a subset of the solution space $\mathscr{S}$, consisting of the solutions that are considered correct. We say a family of SDPs,* $\{SDP^{(p)}\}_{p\in\mathscr{D}}$, solves the problem via the dual *if there is an $\varepsilon \geq 0$ and a function g such that for every $p \in \mathscr{D}$ and every ε-optimal dual-feasible vector $y^{(p)}$ to $SDP^{(p)}$:*

$$g\left(y^{(p)}\right) \in f(p).$$

In other words, an ε-optimal dual solution can be converted into a correct solution of the original problem without more knowledge of p.

For these kinds of SDP families we will prove a lower bound on the sparsity of the dual solutions. The idea for this bound is as follows. If you have a lot of different instances that require different solutions, but the SDPs are equivalent up to permuting the constraints and applying basis transformations on $\mathbb{R}^n$, then a dual solution y should have a lot of unique permutations (i.e., $\{\pi(y) : \pi \in S_m\}$ should be large) and hence cannot be too sparse.

Theorem 9.4 *Consider a problem defined by $f : \mathscr{D} \to \mathscr{P}(\mathscr{S})$, and a family of SDPs as in Definition 9.3. Let $\mathscr{T} \subseteq \mathscr{D}$ be such that for all $p, q \in \mathscr{T}$:*

- *$f(p) \cap f(q) = \emptyset$. That is, a solution to p is not a solution to q and vice versa.*
- *The number of constraints m, and the primal variable size n, are the same for $SDP^{(p)}$ and $SDP^{(q)}$.*
- *Let $\left\{A_j^{(p)}\right\}$ be the constraints of $SDP^{(p)}$ and $\left\{A_j^{(q)}\right\}$ those from $SDP^{(q)}$ (and define $C^{(p)}$, $C^{(q)}$, $b_j^{(p)}$, and $b_j^{(q)}$ in the same manner). There exist a unitary $U \in \mathbb{C}^{n\times n}$ and a $\pi \in S_m$ s.t. $U^\dagger A_{\pi(j)}^{(p)} U = A_j^{(q)}$ (and $U^\dagger C^{(p)} U = C^{(q)}$, $b_{\pi(j)}^{(p)} = b_j^{(q)}$). That is, the SDPs are the same up to permutations of the labels of the constraints, and up to basis transformations.*

If $y^{(p)}$ is an ε-optimal dual-feasible vector to $SDP^{(p)}$ for some $p \in \mathscr{T}$, then $y^{(p)}$ is at least $\frac{\log(|\mathscr{T}|)}{\log(m)}$-dense (i.e., has at least that many non-zero entries).

Proof. Let $SDP^{(p)}$ and $SDP^{(q)}$ be as in the lemma. We claim that, up to the permutation of the coordinates of $\mathbb{R}^m$, the feasible regions of the dual problems of the two SDPs are the same. In particular, let $y^{(p)}$ be feasible for the dual of $SDP^{(p)}$ and define

$$y_j^{(q)} := y_{\pi(j)}^{(p)},$$

then

$$0 \preceq \sum_{j=1}^{m} A_{\pi(j)}^{(p)} y_{\pi(j)}^{(p)} - C^{(p)} = \sum_{j=1}^{m} U A_j^{(q)} U^\dagger y_j^{(q)} - U C^{(q)} U^\dagger = U\left(\sum_{j=1}^{m} A_j^{(q)} y_j^{(q)} - C^{(q)}\right) U^\dagger,$$

which implies

$$0 \leq \sum_{j=1}^{m} A_j^{(q)} y_j^{(q)} - C^{(q)}.$$

It follows that $y_j^{(q)}$ is feasible for the dual of $SDP^{(q)}$. With a similar argument the other direction follows as well. If $y^{(p)}$ is an ε-optimal dual-feasible vector of $SDP^{(p)}$, then $y^{(q)}$ is an ε-optimal dual vector for $SDP^{(q)}$, since $\langle y^{(q)}, b^{(q)} \rangle = \langle y^{(p)}, b^{(p)} \rangle$.

Since $f(p) \cap f(q) = \emptyset$ we know that $g\big(y^{(p)}\big) \neq g\big(y^{(q)}\big)$ and so $y^{(p)} \neq y^{(q)}$. Since this is true for every q in $\mathscr{T}$, there should be at least $|\mathscr{T}|$ different vectors $y^{(q)}$ that are permutations of $y^{(p)}$. A k-sparse vector can have k different non-zero entries and hence the number of possible unique permutations of that vector is at most

$$\binom{m}{k} k! = \frac{m!}{(m-k)!} = \prod_{t=m-k+1}^{m} t \leq m^k$$

so

$$\frac{\log(|\mathscr{T}|)}{\log(m)} \leq k. \qquad \square$$

For many applications U will simply be a permutation matrix. The requirements of Theorem 9.4 might seem restrictive at first sight, but for many problems it is natural to assume such a symmetry in the SDP formulation. For example, for graph problems it is often the case that the numbering of the vertices is arbitrary, and that a different labeling should result in a permutation of the SDP and the dual solution.

Example: (s,t)-mincut Consider the (s,t)-mincut problem, i.e., the dual of the (s,t)-maxflow. Specifically, consider a simple set of instance of this problem: all the inputs that are the union of two complete graphs of size $z+1$, where s is in one subgraph, t is in the other, and the other vertices are named $\{1,2,\ldots,2z\}$. Every assignment of the names over the two halves gives a different instance with a unique mincut (in terms of which names fall on which side of the cut). There is exactly one partition of the vertices in two sets that cuts no edges (namely the partition consists of the two complete graphs), and every other partition cuts at least z edges. Hence a $z/2$-approximate cut is a mincut. This means that there are $\binom{2z}{z}$ instances that require a different output. So for every family of LPs (or SDPs) that is symmetric under permutation of the vertices and for which a $z/2$-approximate dual solution gives an (s,t)-mincut, the sparsity of a $z/2$-approximate dual solution is at least[1]

$$\frac{\log\left(\binom{2z}{z}\right)}{\log(m)} \geq \frac{z}{\log(m)},$$

where we used that $\binom{2z}{z} \geq \frac{2^{2z}}{2\sqrt{z}}$.

Let us now quickly discuss what this implies for our SDP-solvers.

[1] Here m is the number of constraints, not the number of edges in the graph.

Lemma 9.5 *If, for some specific SDP of the form* (2.4)*, every ε-optimal dual-feasible vector has at least ℓ non-zero entries, then our upper bound on SDP-solving in the quantum operator input model (with parameter α) from Theorem 4.34 is at least*

$$\widetilde{\Omega}\left(\left(\sqrt{n\ell}+\sqrt{m}\right)\alpha\ell^2\right).$$

Proof. The vector $\bar{y}$ returned by Meta-Algorithm 4.1 is the average of T vectors $y^{(t)}$ that are all 3-sparse, plus one extra 1-sparse term of $\frac{\varepsilon}{R}e_1$, and hence $\ell \leq 3T+1$. This implies that $\gamma = \Omega\left(\sqrt{\ell/\ln(n)}\right)$ by combining the above with $T = \mathcal{O}\left(\gamma^2\ln(n)\right)$. The lower bound from the lemma follows. □

This lemma shows that our SDP-solvers can only give an $o(n+m)$ upper bound when approximate dual solutions can be $\sqrt[4]{n+m}$-sparse. As a final note we mention that our LP-solvers give sparse primal *and* dual solutions. Therefore, a similar argument as above applies with ℓ equal to the minimal sparsity of both the primal and dual solutions.

9.2 | Lower bounds on the quantum query complexity

The previous section discussed the limitations of specific LP and SDP-solving methods, in this section we will prove general quantum query lower bounds that apply to all methods. The lower bounds will all address different settings and input models. Many of the lower bounds for SDP-solving will follow from a lower bound on LP-solving. It is important to note that the dense input model for LPs corresponds with the sparse matrix input model for SDPs. We prove:

- A lower bound of $\Omega\left(\sqrt{n}+\sqrt{m}\right)$ quantum queries for LP-solving in the dense input model for LPs with a constant γ parameter. This lower bound shows that the n and m-dependence of our LP-solver and SDP-solvers is optimal. This lower bound was first shown by Brandão and Svore [BS17].

- A lower bound of $\Omega\left(\sqrt{ns}\right)$ quantum queries in the sparse matrix input model for SDPs, where s is the maximum row and column sparsity as in Chapter 4, with $m=1$ and a constant γ parameter. This lower bound shows that a dependence on s is necessary.

- A lower bound of $\Omega(nm)$ quantum queries for LP-solving in the dense input model when $n=\Theta(m)$ with a γ parameter that is $\Theta\left(n^4\right)$. This shows that a polynomial dependence on γ is necessary to obtain an $o(nm)$ dependence on m and n.

- A lower bound of $\Omega(n)$ quantum queries for obtaining a solution to a non-negative LP up to a constant multiplicative error (here $n=\Theta(m)$). This matches the classical upper bound for the parameters of the specific problem we consider, and hence shows that there is no general speedup in terms of n and m for non-negative LP solving.

- A lower bound of $\Omega\left(\left(\sqrt{m}+\sqrt{n}\right)\frac{\tau}{\varepsilon}\right)$ queries in the Hamiltonian input model with parameter τ (see Section 4.5.1). This implies an $\widetilde{\Omega}\left(\left(\sqrt{m}+\sqrt{n}\right)\frac{\alpha}{\varepsilon}\right)$ lower bound in the operator input model, and allows us to conclude that our dependence on α is optimal.

- A lower bound of $\widetilde{\Omega}\left(\sqrt{m}\frac{B}{\varepsilon}\right)$ queries in the quantum state input model (see Section 4.5.1). This matches our B-dependence, and the m and n-dependence of [AG19a].

9.2.1 | Dependence on the size in the sparse matrix input model

The first lower bound on quantum SDP-solving was given by Brandão and Svore [BS17]. They showed via a reduction from a search problem that there exists an LP with $\gamma = \mathcal{O}(1)$ that requires $\Omega\left(\sqrt{n}+\sqrt{m}\right)$ quantum queries to solve, which implies a lower bound on SDP-solving.

Theorem 9.6 : [BS17] *Let $n, m \geq 1$. There exists an LP (and hence an SDP) with dimension n and with m constraints (and with $R = 1$ and $r = 1$), such that approximating* OPT *up to additive error* $1/3$ *(with bounded error probability) requires* $\Omega\left(\sqrt{n}+\sqrt{m}\right)$ *quantum queries in the dense input model for LPs (and hence in the sparse matrix input model for SDPs).*

Proof. Let $k = \max\{n, m\}$. Let $v \in \{0,1\}^k$ be an instance of a search problem that asks us to determine whether $|v| = 0$ or $|v| = 1$. As discussed in Chapter 6, solving this search problem requires $\Omega\left(\sqrt{k}\right)$ quantum queries to the input string.

If $m \leq n$ then we construct an LP as follows: let $c = v$, and only include the constraint $\sum_{i=1}^{n} x_i \leq 1$. Clearly $R = 1$ for this LP. Furthermore, the dual only has one variable, y_1, which in the optimum will be equal to the optimal value, so $r =$ OPT. The optimal value is 0 if $|v| = 0$ and 1 if $|v| \geq 1$, and hence solving this LP with additive error $1/3$ solves the search problem.

If $n \leq m$ then we simply use the dual of this LP. Finally we note that in both cases the LP can be padded with zeros in order to make the dimensions match the parameters from the lemma. □

Note that this also proves an $\Omega\left(\sqrt{s}\right)$ quantum query lower bound on LP-solving using the sparse input model for LPs, since we can embed an $(s \times s)$-sized dense LP in a larger $(m \times n)$-sized s-sparse LP.

9.2.2 | Dependence on the sparsity of an SDP

In the previous section we obtained a lower bound on SDP-solving via a lower bound on LP-solving. In this section we give a lower bound specifically on SDP-solving by considering non-diagonal constraint matrices. We will show an $\Omega(\sqrt{ns})$ quantum query lower bound in the sparse matrix input model for SDPs with a constant γ parameter. As mentioned in footnote 13 on page 86, our upper bound can be improved to match the $\sqrt{s}$-dependence of this lower bound [AG19c]. This does leave the open question whether an $\Omega(\sqrt{ms})$ lower bound can be proven as well. The work in this section is not yet published.

Lemma 9.7 *Let $n \geq 1$, and let $s \in [n]$ be odd. Approximating the largest eigenvalue (up to additive error $1/3$ and with bounded error probability) of an s-sparse symmetric matrix $C \in \mathbb{R}^{n\times n}$ with $\|C\| \leq 1$, requires $\Omega(\sqrt{ns})$ quantum queries to a sparse matrix input oracle for C in the worst case.*

Proof. Let $k = \frac{s+1}{2}$. Let $v \in \{0,1\}^{(n-k)\times k}$ be an instance of a search problem that asks us to determine whether $|v| = 0$ or $|v| = 1$ using queries to the entries of v. Solving this search problem requires $\Omega\left(\sqrt{nk}\right) = \Omega(\sqrt{ns})$ quantum queries to v.

Now we let $C \in \mathbb{R}^{n\times n}$ be defined by (where we index starting from 0)

$$C_{ij} = \begin{cases} v_{i,j-i} & \text{if } i \leq j < i+k < n \\ v_{j,i-j} & \text{if } i-k < j < i < n-k \\ 0 & \text{else.} \end{cases}$$

The first clause says that a row of v is placed in each row of C (except in the last k rows), starting from the diagonal and going to the right (i.e. $C_{i,i} = v_{i,0}, C_{i,i+1} = v_{i,1}$, and so on). The second clause ensures that C is symmetric.

If $v = 0$ then all the entries in C are zero and hence $\lambda_{\max}(C) = 0$ and $\|C\| = 0$. If $|v| = 1$, then there is an entry in C that is 1, and hence $\lambda_{\max}(C) \geq 1$. In fact, if $|v| = 1$, then $\|C\| = 1$. We conclude that finding the largest eigenvalue of C up to a $1/3$ error solves the search problem.

From the definition of C it should be clear that there are at most $2k-1 = s$ entries in each row and column of C that could be non-zero. We can give an s-sparse oracle for C by simply returning the locations the possible non-zero entries, i.e., the locations where we have placed the bits of v. A value query for C can clearly be implemented using $\mathcal{O}(1)$ queries to an oracle for v.

Note that our C is not actually s-sparse, it is in fact 1-sparse. If we want that all the values of positions that our sparse oracle returns are indeed non-zero, then we can simply change the zero-valued entries that are returned to $\frac{1}{8n}$, since this changes the eigenvalues by at most $\frac{1}{8}$.[2] □

[2] We would need to scale C slightly to ensure $\|C\| \leq 1$, and we would require a slightly more precise solution.

Theorem 9.8 *Let $n \geq 1$ and $s \in [n]$ odd. There exists an SDP with dimension n and sparsity s (and with $m = 1$, $R = 1$ and $r = 1$), such that approximating* OPT *up to additive error $1/3$ (with bounded error probability) requires $\Omega\big(\sqrt{ns}\big)$ quantum queries in the sparse matrix input model.*

Proof. Let C be an instance from Lemma 9.7. Consider the following SDP

$$\begin{aligned} \max_{X \in \mathbb{R}^{n\times n}} \quad & \operatorname{Tr}(CX) \\ \text{s.t.} \quad & \operatorname{Tr}(X) \leq 1 \\ & X \succeq 0. \end{aligned}$$

Clearly $R = 1$ for this SDP, and $r = \mathrm{OPT}$ since $b_1 = 1$. It is well known that this SDP approximates the largest eigenvalue of C. To see that this is indeed true, let x be an eigenvector of C corresponding to the largest eigenvalue. Now let $X := xx^T$, then X is feasible, and X is also optimal since $\operatorname{Tr}\big(Cxx^T\big) = x^T Cx = \lambda_{\max}(C)$. It follows that solving the SDP with $\varepsilon = 1/3$ approximates the largest eigenvalue of C up to error $1/3$, and hence that this SDP requires $\Omega\big(\sqrt{ns}\big)$ quantum queries to solve in the sparse matrix input model. □

9.2.3 | Dependence on γ in the sparse matrix input model

In this section we will show that every LP-solver in the dense input model (and hence every SDP-solver in the sparse input model) that can distinguish two integer optimal values with bounded error probability requires $\Omega\big(\sqrt{\max\{n,m\}}(\min\{n,m\})^{3/2}\big)$ quantum queries in the worst case. At first sight this might seem to conflict with our upper bounds from Chapter 4. However, the LP that we construct here will have a γ parameter that is polynomial in n and m. In fact, our lower bound shows that if $m = \Theta(n)$, then a polynomial dependence on γ is required in order to get an algorithm with an $o(nm)$ dependence on n and m. We will give a reduction from a composition of Majority and OR functions.

Definition 9.9 : Promise Majority-OR-Majority problem *Let* MAJ_k *be the Majority function on k bits and let* OR_k *be the OR function on k bits. Given input bits $Z_{ij\ell} \in \{0,1\}^{a\times b\times c}$ the problem of computing*

$$\begin{aligned} & MAJ_a(\\ & \quad OR_b(MAJ_c(Z_{111},\ldots,Z_{11c}),\ldots,MAJ_c(Z_{1b1},\ldots,Z_{1bc})), \\ & \quad \ldots, \\ & \quad OR_b(MAJ_c(Z_{a11},\ldots,Z_{a1c}),\ldots,MAJ_c(Z_{ab1},\ldots,Z_{abc})) \\ & \quad) \end{aligned}$$

with the promise that

- *Each inner* MAJ_c *is a boundary case, in other words* $\sum_{\ell=1}^{c} Z_{ij\ell} \in \{c/2, c/2+1\}$ *for all* i, j*.*
- *The outer* MAJ_a *is a boundary case, in other words, if* $\tilde{Z} \in \{0,1\}^a$ *is the bitstring that results from all the OR functions, then* $|\tilde{Z}| \in \{a/2, a/2+1\}$*.*

is called the promise MAJ_a-OR_b-MAJ_c *problem.*

Lemma 9.10 *It takes at least* $\Omega\left(a\sqrt{b}\,c\right)$ *quantum queries to the input to solve the promise* MAJ_a-OR_b-MAJ_c *problem with bounded error probability.*

Proof. The promise version of MAJ_k is known to require $\Omega(k)$ quantum queries. Likewise, it is known that the OR_k function requires $\Omega(\sqrt{k})$ queries. Furthermore, the adversary bound tells us that query complexity is multiplicative under composition of functions; Kimmel [Kim13, Lem. A.3 (Lem. 6 on arXiv)] showed that this even holds for promise functions. □

Lemma 9.11 *Determining the value*

$$\sum_{i=1}^{a} \max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell}$$

for a Z *from the promise* MAJ_a-OR_b-MAJ_c *problem up to additive error* $\varepsilon = 1/3$*, solves the promise* MAJ_a-OR_b-MAJ_c *problem.*

Proof. Notice that due to the first promise, $\sum_{\ell=1}^{c} Z_{ij\ell} \in \{c/2, c/2+1\}$ for all $i \in [a]$ and $j \in [b]$. This implies that

- If the ith OR is 0, then all of its inner MAJ functions evaluate to 0 and hence
$$\max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell} = \frac{c}{2}.$$
- If the ith OR is 1, then at least one of its inner MAJ functions evaluates to 1 and hence
$$\max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell} = \frac{c}{2} + 1.$$

Now, if we denote the string of outcomes of the OR functions by $\tilde{Z} \in \{0,1\}^a$, then

$$\sum_{i=1}^{a} \max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell} = a\frac{c}{2} + \left|\tilde{Z}\right|.$$

Hence determining the left-hand side will determine $\left|\tilde{Z}\right|$; this Hamming weight is either $\frac{a}{2}$ if the full function evaluates to 0, or $\frac{a}{2} + 1$ if it evaluates to 1. □

Lemma 9.12 *For an input $Z \in \{0,1\}^{a\times b\times c}$ there is an LP with $m = c+a$ and $n = c+ab$ for which the optimal value is*

$$\sum_{i=1}^{a} \max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell}.$$

Furthermore, a query in the dense input model for this LP can be implemented with 1 *query to Z.*

Proof. Let $Z^{(i)}$ be the matrix one gets by fixing the first index of Z and putting the entries in a $c\times b$ matrix, so $Z^{(i)}_{\ell j} = Z_{ij\ell}$. We define the following LP:

$$\begin{aligned} \text{OPT} = \max \quad & \sum_{k=1}^{c} w_k \\ \text{s.t.} \quad & \begin{bmatrix} I & -Z^{(1)} & \cdots & -Z^{(a)} \\ 0 & \mathbf{1}^T & & \\ \vdots & & \ddots & \\ 0 & & & \mathbf{1}^T \end{bmatrix} \begin{bmatrix} w \\ v^{(1)} \\ \vdots \\ v^{(a)} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\ & v^{(1)},\ldots,v^{(a)} \in \mathbb{R}^b_{\geq 0}, w \in \mathbb{R}^c_{\geq 0}. \end{aligned}$$

Notice that every $Z^{(i)}$ is of size $c\times b$, so that indeed $m = c+a$ and $n = c+ab$.

For every $i \in [a]$ there is a constraint that says

$$\sum_{j=1}^{b} v^{(i)}_j \leq 1.$$

The constraints involving w say that for every $k \in [c]$

$$w_k \leq \sum_{i=1}^{a}\sum_{j=1}^{b} v^{(i)}_j Z^{(i)}_{kj} = \sum_{i=1}^{a} (Z^{(i)} v^{(i)})_k,$$

where $(Z^{(i)}v^{(i)})_k$ is the kth entry of the matrix-vector product $Z^{(i)}v^{(i)}$. Clearly, for an optimal solution these constraints will be satisfied with equality, since in the objective function w_k has a positive weight, and each w_j is only part of a single constraint. Summing over k on both sides, we get the equality

$$\begin{aligned} \text{OPT} &= \sum_{k=1}^{c} w_k \\ &= \sum_{k=1}^{c}\sum_{i=1}^{a} (Z^{(i)}v^{(i)})_k \\ &= \sum_{i=1}^{a}\sum_{k=1}^{c} (Z^{(i)}v^{(i)})_k \\ &= \sum_{i=1}^{a} \left\| Z^{(i)} v^{(i)} \right\|_1, \end{aligned}$$

so in the optimum $\left\|Z^{(i)}v^{(i)}\right\|_1$ will be maximized. Note that we can use the ℓ_1-norm as a shorthand for the sum over vector elements since all elements of $Z^{(i)}v^{(i)}$ are positive. In particular, the value of $\left\|Z^{(i)}v^{(i)}\right\|_1$ is given by

$$\begin{aligned} \max \quad & \left\|Z^{(i)}v^{(i)}\right\|_1 \\ \text{s.t.} \quad & \left\|v^{(i)}\right\|_1 \leq 1 \\ & v^{(i)} \geq 0. \end{aligned}$$

Now $\|Z^{(i)}v^{(i)}\|_1$ will be maximized by putting all weight in $v^{(i)}$ on the index that corresponds to the column of $Z^{(i)}$ that has the highest Hamming weight. In particular, in the optimum $\|Z^{(i)}v^{(i)}\|_1 = \max_{j\in[b]}\sum_{\ell=1}^{c} Z^{(i)}_{\ell j}$. Putting everything together gives:

$$\mathrm{OPT} = \sum_{i=1}^{a}\left\|Z^{(i)}v^{(i)}\right\|_1 = \sum_{i=1}^{a}\max_{j\in[b]}\sum_{\ell=1}^{c} Z^{(i)}_{\ell j} = \sum_{i=1}^{a}\max_{j\in[b]}\sum_{\ell=1}^{c} Z_{ij\ell}.$$

□

Theorem 9.13 *Let $n, m \geq 3$. There exists an LP (and hence an SDP) with dimension n and with m constraints (and with $R, r = \mathcal{O}(\min\{n,m\}^2)$), such that approximating* OPT *up to additive error* $1/3$ *(with bounded error probability) requires*

$$\Omega\left(\sqrt{\max\{n,m\}}(\min\{n,m\})^{3/2}\right)$$

quantum queries in the dense input model for LPs (and hence in the sparse matrix input model for SDPs with $s = 1$).

Proof. We start with the case where $m \leq n$. We first assume that there exist $a \geq 1$ and $k \geq 2$ such that $m = 2a$ and $n = ka$. Let $c = a = m/2$ and $b = 2k = \frac{n-c}{a} = \frac{2n}{m} - 1$, so that $n = c + ab$ and $m = c + a$. By Lemma 9.12 there exists an LP with $n = c + ab$ and $m = c + a$ that computes

$$\sum_{i=1}^{a}\max_{j\in[b]}\sum_{\ell=1}^{c} Z_{ij\ell}$$

for an input Z to the promise MAJ_a-OR_b-MAJ_c problem. By Lemma 9.11, computing this value will solve the promise MAJ_a-OR_b-MAJ_c problem. By Lemma 9.10 the promise MAJ_a-OR_b-MAJ_c problem takes $\Omega(a\sqrt{b}c)$ quantum queries in the worst case. This implies a lower bound of

$$\Omega\left(m^2\sqrt{\frac{n}{m}}\right) = \Omega(m^{3/2}\sqrt{n})$$

quantum queries on solving these LPs.

For the upper bound on R note that for an optimal point $\|w\|_1 = \mathrm{OPT}$ and $\sum_{i=1}^{a}\left\|v^{(i)}\right\|_1 = a$. Hence

$$R = \mathrm{OPT} + a \leq \frac{ac}{2} + \frac{a}{2} + 1 + a = \mathcal{O}(ac) = \mathcal{O}\left(m^2\right).$$

For the upper bound on r we consider the dual of the LP:

$$\begin{aligned} \mathrm{OPT} = \min \quad & \sum_{i=1}^{a} y_i \\ \text{s.t.} \quad & \begin{bmatrix} I & 0 & \cdots & 0 \\ -Z^{(1)} & \mathbf{1} & & \\ \vdots & & \ddots & \\ -Z^{(a)} & & & \mathbf{1} \end{bmatrix} \begin{bmatrix} \phi \\ y_1 \\ \vdots \\ y_a \end{bmatrix} \geq \begin{bmatrix} \mathbf{1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ & \phi \in \mathbb{R}^c_{\geq 0}, y \in \mathbb{R}^a_{\geq 0}. \end{aligned}$$

Clearly all entries of ϕ are going to be as small as possible for an optimal point, so $\phi = \mathbf{1}$ and $\|\phi\|_1 = c$. It is easy to see that for an optimal point $\|y\|_1 = \mathrm{OPT}$, so

$$r = \mathrm{OPT} + c \leq \frac{ac}{2} + \frac{a}{2} + 1 + c = \mathcal{O}(ac) = \mathcal{O}\left(m^2\right).$$

Now, if m is odd, then we let $a = \frac{m-1}{2}$ and add a dummy constraint to the LP. If n is not a multiple of a then we let $b = \lfloor \frac{n}{a} \rfloor$ and add $n - c - ab$ dummy variables to the LP. Since $n \geq m = 2a$ we find that $b \geq 1$. Finally, the case where $n \leq m$ follows from duality. □

It is important to note that the parameters R and r are not constant for the LP from Theorem 9.13, and hence this lower bound does not contradict the scaling with $\sqrt{m} + \sqrt{n}$ of the complexity of our LP-solver (see Theorem 5.13). The lower bound implies that any algorithm with an $o(mn)$ dependence on n and m has to depend at least polynomially on $\gamma = Rr/\varepsilon$. For example, the lower bound shows that an algorithm with a $\sqrt{m} + \sqrt{n}$ dependence has to have an γ^κ factor in its complexity with $\kappa \geq 3/8$. It remains an open question whether a tighter lower bound on the error dependence can be proven.[3]

9.2.4 | Non-negative LP-solvers

A non-negative LP (NNLP) is an LP of the form

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} x_i \\ \text{s.t.} \quad & Ax \leq \mathbf{1} \\ & x \geq 0. \end{aligned}$$

[3] For LP-solvers with a $\sqrt{m} + \sqrt{n}$ dependence on n and m a slightly better lower bound of $\kappa \geq 1/2$ follows from Theorem 9.15. However, in other settings Theorem 9.15 will not always give a lower bound on κ. For example, Theorem 9.15 will not give any such lower bound for LP-solvers with a $\sqrt{nm}$-dependence.

where $A \in \mathbb{R}_{\geq 0}^{m\times n}$ is entrywise non-negative. Many of the algorithms for this type of LP assume access to A in the form of a single list of indices that includes the indices of all non-zero entries. The notation $\mathrm{nnz}(A)$ is used to denote the length of this list. We call this input model the *entrywise input model.* An important result about NNLPs by Luby and Nisan [LN93] is that there exists a classical algorithm that approximates OPT up to *multiplicative* error δ in $\widetilde{\mathcal{O}}_{nm}\left(\frac{\mathrm{nnz}(A)}{\delta^4}\right)$ steps. The main point here is the lack of a dependence on R and r. Their results have later been improved in many different settings, including by Koufogiannakis and Young [KY14] to $\widetilde{\mathcal{O}}_{nm}\left(\frac{n+m}{\delta^2}+\mathrm{nnz}(A)\right)$ for sequential algorithms (there are also many results in parallel or distributed settings).

In this section we give a family of NNLPs with $n = \Theta(m) = \Theta(\mathrm{nnz}(A))$, for which $\Omega(n)$ quantum queries to a dense input oracle for A, or to an entrywise input oracle for A, are needed in order find a primal solution that has objective value at least $(1-\delta)$OPT. The work in this section is not yet published.

Lemma 9.14 *There exists a constant* ε_0 *such that for all* $\varepsilon \in (0, \varepsilon_0)$ *the following holds: given an oracle for a string* $z \in \{0,1\}^n$ *with* $|z| \notin \{0, n\}$*, finding a string* $\tilde{z}$ *(with probability at least* $5/6$*) such that* $|z - \tilde{z}| \leq \varepsilon n$*, requires* $\Omega(n)$ *quantum queries to the entries of* z *in the worst case.*

Proof. Assume $\mathscr{A}$ is a quantum algorithm that uses T quantum queries to z and with probability at least $5/6$ finds a $\tilde{z}$ such that $|z-\tilde{z}| \leq \varepsilon n$. We will construct a bounded-error quantum algorithm that recovers z fully. It is known that such an algorithm requires at least $n/2-2$ queries to the input, since it allows us to compute the parity of z [BBCMW01].[4]

Boyer et al. [BBHT98] showed that Grover's algorithm can be used to find one out of t solutions in a search space of size n using an expected number of $\mathcal{O}\left(\sqrt{n/t}\right)$ quantum queries, even if we do not know t. Let c be the constant hidden in the big-O notation, so $c\sqrt{n/t}$ quantum queries suffice in expectation. Then, to find all t solutions

$$c\sum_{i=0}^{t-1}\sqrt{\frac{n}{t-i}} = c\sum_{i=1}^{t}\sqrt{\frac{n}{i}} \leq c\int_0^t \sqrt{\frac{n}{x}}dx = 2c\sqrt{nt}$$

quantum queries suffice in expectation, since we can exclude already-found solutions from subsequent searches.

Now to reconstruct z fully. We first apply $\mathscr{A}$ (using T queries) to find a $\tilde{z}$ such that $|z-\tilde{z}| \leq \varepsilon n$ with probability at least $5/6$. Let $x = z \oplus \tilde{z}$ be the string of differences, so $|x| \leq \varepsilon n$. Clearly we can make queries to x given access to quantum queries to z. We can now use at most $2c\sqrt{n\cdot\varepsilon n} = 2c\sqrt{\varepsilon}n$ quantum queries in expectation to find all differences between z and $\tilde{z}$, and hence reconstruct z. We can modify the search algorithm to stop after $12c\sqrt{\varepsilon}n$ queries, so by Markov's inequality it finds a correct z with probability at least $5/6$.

[4] Note that we do not get an $n/2$ lower bound since we excluded the all-ones and all-zeros strings.

Now, with probability at least $(5/6)^2 \geq 2/3$, we can learn z using $T + 12c\sqrt{\varepsilon}n$ queries. Since at least $n/2 - 2$ queries are required to learn z, it follows that

$$T \geq \left(\frac{1}{2} - 12c\sqrt{\varepsilon}\right)n - 2.$$

So by letting $\varepsilon_0 < \frac{1}{576c^2}$ we get the stated lower bound. □

Theorem 9.15 *Let ε_0 be the constant from Lemma 9.14. Let $n \geq 4/\varepsilon_0$, and let $0 < \delta < \varepsilon_0/10$. There exists a non-negative LP with dimension n, with $m = n+1$ constraints, and with at most $2n$ fixed positions in the constraint matrix that might be non-zero, such that approximating* OPT *up to multiplicative error δ (with bounded error probability) requires $\Omega(n)$ quantum queries in the dense input model for LPs, as well as in the entrywise input model for LPs.*

Proof. Let $\varepsilon = \frac{2}{n} + 5\delta < \varepsilon_0$. Let $z \in \{0,1\}^n$ be an input from Lemma 9.14 for this ε, so $|z| \notin \{0, n\}$. Consider the following non-negative LP:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} x_i \\ \text{s.t.} \quad & \langle z, x\rangle \leq 1 \\ & x_i \leq 1 \text{ for all } i \in [n] \\ & x \geq 0. \end{aligned}$$

It is easy to see that a dense input oracle for this LP can easily be implemented using a single query to an oracle for z. Furthermore, the number of positions in the constraint matrix that can be non-zero is $2n$, and these positions can easily be listed. Let x^* be an optimal solution. Clearly $x_i^* = 1$ for all i where $z_i = 0$. Furthermore, $\sum_{i:z_i=1} x_i \leq 1$. Hence $\mathrm{OPT} = |\bar{z}| + 1$, where $\bar{z}$ is the entrywise negation of z.

Now let $\tilde{x}$ be a feasible point with optimal value at least $(1-\delta)\cdot\mathrm{OPT}$. Then

$$\sum_{i=1}^{n} \tilde{x}_i \geq (1-\delta)\mathrm{OPT} = (1-\delta)(|\bar{z}|+1).$$

Since $\tilde{x}$ is feasible we know that

$$\sum_{i:z_i=1} \tilde{x}_i \leq 1,$$

hence there are at most two indices i such that $z_i = 1$ and $\tilde{x}_i \geq 1/2$. It also follows that

$$\sum_{i:z_i=0} \tilde{x}_i \geq (1-\delta)(|\bar{z}|+1) - 1 \geq (1-2\delta)|\bar{z}|.$$

Hence, for at least $(1-5\delta)|\bar{z}|$ of the i where $z_i = 0$ we have $\tilde{x}_i > 1/2$, and for at most $5\delta|\bar{z}| \leq 5\delta n$ of these i we have $\tilde{x}_i \leq 1/2$. Let $\tilde{z}$ be the string we get by rounding the entries of $\tilde{x}$ to $\{0,1\}$ and then flipping all the bits. Then

$$|z - \tilde{z}| \leq 2 + 5\delta n = \varepsilon n.$$

We conclude that, due to Lemma 9.14, at least $\Omega(n)$ quantum queries in the dense input model are needed to solve the non-negative linear program. Similarly $\Omega(n)$ quantum queries in the entrywise input model with parameter $\mathrm{nnz}(A) = 2n$ are needed. □

Due to duality we know that the same lower bound holds for finding a dual solution. As mentioned at the beginning of this section, some classical algorithms already run in time $\widetilde{\mathcal{O}}\left(\frac{n+m}{\varepsilon^2} + \mathrm{nnz}(A)\right)$, which is $\mathcal{O}\left(\frac{n}{\varepsilon^2}\right)$ for the parameters of the LP from Theorem 9.15. This shows that, apart from a possible improvement in the error dependence, there is no unconditional quantum speed-up possible for non-negative linear programming.

One area in which a quantum speedup is still possible would be approximating only the optimal *value*, without also giving an optimizing *point*. For such an algorithm the LP above would not give an $\Omega(n)$ lower bound, since approximating the Hamming weight of a string up to a multiplicative error δ can be done using $\mathcal{O}(\sqrt{n}/\delta)$ queries [BHT98]. Another approach for finding a quantum speed-up could be to look for an algorithm that is fast when $m \ll n$ (or the other way around), for example, an algorithm with a $\sqrt{nm}$ complexity. Finally, a quantum algorithm might get a better bound for dense matrices, by improving (or removing) the dependence on $\mathrm{nnz}(A)$.

9.2.5 | The Hamiltonian & quantum operator input models

We now modify the lower bound from Theorem 9.6 to prove a lower bound in the Hamiltonian input model. This lower bound will also imply a lower bound in the quantum operator model.

Lemma 9.16 *Let $\tau \geq 1$ and $\delta \in (0,1]$. Let $x \in \{0,1\}^n$ such that $|x| = 0$ or $|x| = 1$. Given access to an oracle acting as*

$$O_x|i\rangle = e^{\mathbf{i}\left(\frac{1}{2\tau}+\frac{\delta x_i}{\tau}\right)}|i\rangle,$$

determining (with probability 2/3) which of the cases for x holds requires at least $\Omega\left(\sqrt{n}\frac{\tau}{\delta}\right)$ queries to O_x.

Proof. This follows from the Phase Adversary Method (Lemma 6.4) with a similar argument to the example in Section 6.4.2. □

We now reprove Theorem 9.6 for the Hamiltonian input model. Remember that in this model (with parameter τ) we access A_j via a unitary $e^{\mathbf{i}A_j/t_j}$ with $t_j \leq \tau$ (see Section 4.5.1 for details).

Theorem 9.17 *Let $n, m \geq 1, \tau \geq 1$ and $\varepsilon \in (0,1/3]$. There exists an SDP with dimension n and with m constraints (and with $R, r = \mathcal{O}(1)$), such that approximating* OPT *up to additive error ε (with probability at least 5/6) requires $\Omega\left(\left(\sqrt{n}+\sqrt{m}\right)\frac{\tau}{\varepsilon}\right)$ queries in the Hamiltonian input model with parameter τ.*

Proof. Let $k = \max\{n, m\}$. Let $v \in \{0,1\}^k$ be an instance of the problem in Lemma 9.16, with $\delta = 3\varepsilon$.

If $m \leq n$ then we define an SDP as follows: let $C = I/2 + \mathrm{diag}(\delta v)$, and only include the constraint $\mathrm{Tr}(X) \leq 1$. We let the parameters of the Hamiltonian input model be $t_0 = t_1 = \tau$. As in Theorem 9.6 we have $R, r = \mathcal{O}(1)$. A query in the Hamiltonian input model for this SDP can be implemented using a single phase query to the oracle from Lemma 9.16. Finally we note that $\mathrm{OPT} = 1/2$ if $|v| = 0$, and that $\mathrm{OPT} = 1/2 + \delta$ if $|v| = 1$. Hence, approximating OPT with error ε solves the search problem from Lemma 9.16, and therefore requires $\Omega\big(\sqrt{n}\frac{\tau}{\varepsilon}\big)$ queries.

If $n \leq m$ then consider the dual of this SDP. Since the input matrices are diagonal, this dual can again be written as a primal problem by negating both sides of each constraint. The dual has a single variable and the queries to the Hamiltonian input model for this SDP will again be implementable with a single query to the oracle from Lemma 9.16.

Finally we note that we can add dummy constraints or variables in order to make the dimensions of the problem match n and m. □

Using Lemma 4.25 we get the following result in the quantum operator input model.

Corollary 9.18 *Let $n, m \geq 1$, $\alpha \geq \frac{4}{\pi}$ and $\varepsilon \in (0, 1/3]$. There exists an LP (and hence SDP) with dimension n and with m constraints (and with $R, r = \mathcal{O}(1)$), such that approximating* OPT *up to additive error ε (with probability* 5/6*) requires $\widetilde{\Omega}\big((\sqrt{n} + \sqrt{m})\frac{\alpha}{\varepsilon}\big)$ queries in the quantum operator input model with parameter* α.

Proof. Let $\mathscr{A}$ be a T-query algorithm that approximates OPT up to additive error ε in the operator input model with probability $5/6$. Consider an input from Theorem 9.17 for $\tau \geq 2$. If Lemma 4.25 would work without error then we could implement a query to the operator input model for this SDP using $\mathcal{O}(1)$ queries to the Hamiltonian input model for this SDP. It would follow that $T = \Omega\big((\sqrt{n} + \sqrt{m})\frac{\alpha}{\varepsilon}\big)$.

However, Lemma 4.25 does introduce error. Let U be the unitary corresponding to $\mathscr{A}$ when we apply the algorithm with perfect queries in the operator input model. Let $\tilde{U}$ be the unitary corresponding to $\mathscr{A}$ if we use Lemma 4.25 to approximate the queries, with the error set to μ. Then

$$\|U - \tilde{U}\| \leq T \cdot \mu,$$

so if we pick $\mu = \mathcal{O}\big(\frac{1}{T}\big)$ then $\mathscr{A}$ finds a solution with probability at least $2/3$. One query in the operator model can then be implemented using $\widetilde{\mathcal{O}}_T(1)$ queries in the Hamiltonian input model. It follows that $T = \widetilde{\Omega}\big((\sqrt{n} + \sqrt{m})\frac{\alpha}{\varepsilon}\big)$. □

9.2.6 | The quantum state input model

We finish this section with a lower bound for the quantum state input model. Remember that in this model (with parameter B) we access A_j via a unitary that prepares purification of subnormalized states ρ_j^+ and ρ_j^-, such that a given linear combination of

ρ_j^+, ρ_j^-, and I is equal to A. Recall that B is an upper bound on the sum of the coefficients of ρ_j^+ and ρ_j^- in this linear combination (see Section 4.5.1 for details). In this model we will not be able to prove an $\Omega(\sqrt{n})$ lower bound due to the upper bounds of [BKLLSW19] and [AG19a].

Lemma 9.19 *Let $\xi \in (0,1/3]$. Assume we are given an SDP in the quantum operator input model with parameter $\alpha \geq 2$. Using $\widetilde{\mathcal{O}}_{\frac{n\alpha}{\xi}}(1)$ queries to the oracle for the quantum operator input model, an oracle for the quantum state input model can be implemented encoding matrices $\tilde{A}_j$ that are ξ-close in trace distance to the original A_j matrices. This new oracle for the quantum state model has a B parameter equal to $128n\alpha$.*

Proof. We consider a single A_j, and for ease of notation we omit the subscript. The proof will draw heavily on the ideas developed for Gibbs-sampling in Lemma 4.15. Let the spectral decomposition of A be

$$A = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle\langle\psi_i|.$$

By assumption we have an $(\alpha, a, 0)$-block-encoding U of A for some number of qubits a. Using Lemma 2.11 we can implement a $\left(1, \mathcal{O}(a), \frac{\xi}{128n\alpha}\right)$-block-encoding of

$$\frac{\sqrt{I + \frac{A}{8\alpha}}}{4}$$

with $\widetilde{\mathcal{O}}_{\frac{n\alpha}{\xi}}(1)$ applications of U using the polynomial Q from Lemma 4.17. Applying this block-encoding to the first half of a maximally entangled state results in a state that is $\frac{\xi}{128n\alpha}$-close in ℓ_2-norm to

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \frac{\sqrt{1 + \frac{\lambda_i}{8\alpha}}}{4} |\psi_i\rangle|\bar{\psi}_i\rangle|0\rangle + |\perp\rangle,$$

where $|\perp\rangle$ has no overlap with the all-zero state in the last register.

Would we trace out the second register, and project on the part of the state with a 0 in the last register, then we would get a subnormalized state that is $\frac{\xi}{128n\alpha}$-close in trace norm to

$$\frac{1}{16n}\left(I + \frac{A}{8\alpha}\right).$$

Now, by letting $\mu^I = -\frac{1}{16n}$ and $\mu^+ = 128n\alpha$ we have implemented a query in the quantum state input model with $B = 128n\alpha$ for an $\tilde{A}$ that is ξ-close to A in trace norm. □

Corollary 9.20 *Let $\varepsilon \in (0,1/6]$, $m \geq 2$, and $B \geq 256$. Then there is an LP (and hence an SDP) dimension $n = 2$ and with m constraints (and with $R, r = \mathcal{O}(1)$) for which finding an ε-approximation of the optimal value (with probability $11/12$) requires $\widetilde{\Omega}\left(\sqrt{m}\frac{B}{\varepsilon}\right)$ queries in the quantum state model with parameter B.*

Proof. This follows from the reduction above with a similar argument as in the proof of Corollary 9.18. □

9.3 | Discussion and future work

The results from Section 9.2 show that our SDP-solving algorithms from Chapter 4 and our LP-solving algorithms from Chapter 5 are optimal, apart from a possible improvement in the γ-dependence. Furthermore, Theorem 9.13 shows that a polynomial dependence on γ is required in order to get an $o(nm)$ upper bound in terms of n and m. Considering that the results from Section 9.1 show that a large γ parameter is natural for many problems, the only way we see that further research into these types of LP and SDP-solvers could yield useful results, is to improve the γ-dependence drastically. For our type of solvers this would require an improvement in the iteration count *and* in the γ-dependence of each iteration.

However, little work has yet been done on quantum LP and SDP-solvers with a *polylogarithmic* error dependence. Classically, cutting plane methods have given very strong theoretical results, and interior point methods are widely used in practice. It would be an interesting direction for future work to consider if these methods can be sped up using quantum algorithms.

Bibliography

[Aar18] S. Aaronson. “Shadow Tomography of Quantum States”. In: *Proceedings of the 50th ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 325–338. arXiv: `1711.01053v1`.

[AB16] S. Aaronson and S. Ben-David. “Sculpting Quantum Speedups”. In: *Proceedings of the 31st Conference on Computational Complexity*. CCC ’16. 2016, pp. 26:1–26:28.

[ABP19] S. Arunachalam, J. Briët, and C. Palazuelos. “Quantum Query Algorithms Are Completely Bounded Forms”. In: *SIAM Journal on Computing* 48.3 (2019), pp. 903–925. arXiv: `1711.07285v2`.

[AG18] J. van Apeldoorn and S. Gribling. “Simon’s problem for linear functions”. 2018. arXiv: `1810.12030v2`.

[AG19a] J. van Apeldoorn and A. Gilyén. “Improvements in Quantum SDP-Solving with Applications”. In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). 2019, pp. 99:1–99:15. arXiv: `1804.05058v1`.

[AG19b] J. van Apeldoorn and A. Gilyén. “Quantum algorithms for zero-sum games”. 2019. arXiv: `1904.03180v1`.

[AG19c] J. van Apeldoorn and S. Gribling. *Unpublished work on more efficient block-encodings for sparse matrices*. 2019.

[AGGW17] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. “Quantum SDP-Solvers: Better upper and lower bounds”. In: *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 403–414. arXiv: `1705.01843v3`.

[AGGW18] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. “Convex optimization using quantum oracles”. 2018. Accepted to Quantum. arXiv: `1809.00643v3`.

[AHK12] S. Arora, E. Hazan, and S. Kale. “The Multiplicative Weights Update Method: a Meta-Algorithm and Applications”. In: *Theory of Computing* 8.6 (2012), pp. 121–164.

[AK16] S. Arora and S. Kale. “A Combinatorial, Primal-Dual Approach to Semidefinite Programs”. In: *Journal of the ACM* 63.2 (2016), pp. 12:1–12:35. Earlier version in STOC’07.

[Amb00] A. Ambainis. “Quantum lower bounds by quantum arguments”. In: *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC).* 2000, pp. 636–643. arXiv: `quant-ph/0002066v1`.

[Amb99] A. Ambainis. “A Better Lower Bound for Quantum Algorithms Searching an Ordered List”. In: *Proceedings of the 40thm IEEE Symposium on Foundations of Computer Science (FOCS).* 1999, pp. 352–357. arXiv: `quant-ph/9902053v1`.

[Aru+19] F. Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pp. 505–510.

[BBCMW01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. “Quantum Lower Bounds by Polynomials”. In: *Journal of the ACM* 48.4 (2001), pp. 778–797. arXiv: `quant-ph/9802049v3`. Earlier version in FOCS’98.

[BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. “Tight Bounds on Quantum Searching”. In: *Fortschritte der Physik* 46.4–5 (1998), pp. 493–505. arXiv: `quant-ph/9605034v1`. Earlier version in Physcomp’96.

[BCK15] D. Berry, A. Childs, and R. Kothari. “Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters”. In: *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS).* 2015, pp. 792–809. arXiv: `1501.01715v3`.

[Bel15] A. Belovs. “Variations on Quantum Adversary”. 2015. arXiv: `1504.06943v1`.

[BHJ26] M. Born, W. Heisenberg, and P. Jordan. “Zur Quantenmechanik II”. In: *Zeitschrift für Physik* 35 (1926), pp. 557–615.

[BHMT02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. “Quantum Amplitude Amplification and Estimation”. In: *Quantum Computation and Quantum Information: A Millennium Volume.* Vol. 305. AMS Contemporary Mathematics Series. 2002, pp. 53–74. arXiv: `quant-ph/0005055v1`.

[BHT98] G. Brassard, P. Høyer, and A. Tapp. “Quantum Counting”. In: *Proceedings of 25th ICALP.* Vol. 1443. Lecture Notes in Computer Science. 1998, pp. 820–831. arXiv: `quant-ph/9805082v1`.

[BJ25] M. Born and P. Jordan. “Zur Quantenmechanik”. In: *Zeitschrift für Physik* 34 (1925), pp. 858–888.

[BKLLSW17] F. Brandão, A. Kalev, T. Li, C. Lin, K. Svore, and X. Wu. “Exponential Quantum Speed-ups for Semidefinite Programming with Applications to Quantum Learning”. 2017. First arXiv version. arXiv: `1710.02581v1`.

[BKLLSW19] F. Brandão, A. Kalev, T. Li, C. Lin, K. Svore, and X. Wu. "Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning". In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). 2019. arXiv: `1710.02581v3`.

[Boh13] N. Bohr. "On the Constitution of Atoms and Molecules". In: *Philosophical Magazine*. 6th ser. 26 (1913), pp. 1–25.

[BS17] F. Brandão and K. Svore. "Quantum Speed-ups for Solving Semidefinite Programs". In: *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 415–426. arXiv: `1609.05537v3`.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge univeristy press, 2004.

[CCLW18] S. Chakrabarti, A. Childs, T. Li, and X. Wu. "Quantum algorithms and lower bounds for convex optimization". 2018. arXiv: `1809.01731v2`.

[CDST19] Y. Carmon, J. Duchi, A. Sidford, and K. Tian. "A Rank-1 Sketch for Matrix Multiplicative Weights". 2019. arXiv: `1903.02675v2`.

[CEMM98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. "Quantum algorithms revisited". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339–354. arXiv: `quant-ph/9708016v1`.

[CGJ18] S. Chakraborty, A. Gilyén, and S. Jeffery. "The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation". 2018. arXiv: `1804.01973v2`.

[Chi10] Andrew M. Childs. "On the relationship between continuous- and discrete-time quantum walk". In: *Communications in Mathematical Physics* 294.2 (2010), pp. 581–603. arXiv: `0810.0312v3`.

[CHTW04] R. Cleve, P. Høyer, B. Toner, and J. Watrous. "Consequences and limits of nonlocal strategies". In: *Proceedings of the 19th IEEE Conference on Computational Complexity (CCC)*. 2004, pp. 236–249. arXiv: `quant-ph/0404076v2`.

[Chu36] A. Church. "An Unsolvable Problem of Elementary Number Theory". In: *American Journal of Mathematics* 58.2 (1936), pp. 345–363.

[CLS19] M. Cohen, Y. Lee, and Z. Song. "Solving linear programs in the current matrix multiplication time". In: *Proceedings of the 51th ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 938–942.

[Dan48] G. Dantzig. "Programming in a Linear Structure". In: *Electronics Magazine* 17 (1948). Report of the September 9, 1948 meeting in Madison, pp. 73–74.

[DH96] C. Dürr and P. Høyer. "A Quantum Algorithm for Finding the Minimum". 1996. arXiv: `quant-ph/9607014v2`.

[DHHM06] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. "Quantum Query Complexity of Some Graph Problems". In: *SIAM Journal on Computing* 35.6 (2006), pp. 1310–1328.

[DJ92] D. Deutsch and R. Jozsa. "Rapid Solution of Problems by Quantum Computation". In: *Proceedings of the Royal Society of London.* Vol. A439. 1992, pp. 553–558.

[Edm65] J. Edmonds. "Paths, trees, and flowers". In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467.

[Ein05] A. Einstein. "Concerning an Heuristic Point of View Toward the Emission and Transformation of Light". In: *Annalen der Physik* 17 (1905), pp. 132–148.

[Eld03] Y. Eldar. "A Semidefinite Programming Approach to Optimal Unambiguous Discrimination of Quantum States". In: *IEEE Transactions on Information Theory* 49 (2003), pp. 446–456. arXiv: `quant-ph/0206093v2`.

[EZ64] H. Ehlich and K. Zeller. "Schwankung von Polynomen zwischen Gitterpunkten". In: *Mathematische Zeitschrift* 86 (1964), pp. 41–44.

[Fey82] R. Feynman. "Simulating Physics with Computers". In: *International Journal of Theoretical Physics* 21.6/7 (1982), pp. 467–488.

[Fey85] R. Feynman. "Quantum Mechanical Computers". In: *Optics News* 11 (1985), pp. 11–20.

[GAW19] A. Gilyén, S. Arunachalam, and N. Wiebe. "Optimizing quantum optimization algorithms via faster quantum gradient computation". In: *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA).* 2019, pp. 1425–1444. arXiv: `1711.00465v3`.

[GK95] M. Grigoriadis and L. Khachiyan. "A Sublinear-time Randomized Approximation Algorithm for Matrix Games". In: *Operations Research Letters* 18.2 (1995), pp. 53–58.

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Combinatorica* 1.2 (1981), pp. 169–197.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer, 1988.

[Gro96] L. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC).* 1996, pp. 212–219. arXiv: `quant-ph/9605043v3`.

[GSLW19] A. Gilyén, Y. Su, G. Low, and N. Wiebe. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In: *Proceedings of the 51st ACM Symposium on Theory of Computing (STOC)*. 2019. arXiv: `1806.01838v1`.

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd ed. SIAM, 2008.

[GW95] M. Goemans and D. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM* 42.6 (1995), pp. 1115–1145. Earlier version in STOC'94.

[Hei25] W. Heisenberg. "Über quantentheoretische Umdeutung kinematischer und mechanischer Beziehungen". In: *Zeitschrift für Physik* 33 (1925), pp. 879–893.

[HLM17] A. Harrow, C. Lin, and A. Montanaro. "Sequential measurements, disturbance and property testing". In: *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2017, pp. 1598–1611. arXiv: `1607.03236v3`.

[HLŠ07] P. Høyer, T. Lee, and R. Špalek. "Negative weights make adversaries stronger". In: *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*. 2007, pp. 526–535. arXiv: `quant-ph/0611054v2`.

[JJUW11] R. Jain, Z. Ji, S. Upadhyay, and J. Watrous. "QIP = PSPACE". In: *Journal of the ACM* 58.6 (2011), pp. 30:1–30:27.

[Jor05] S. Jordan. "Fast Quantum Algorithm for Numerical Gradient Estimation". In: *Physical Review Letters* 95.5 (2005). arXiv: `quant - ph / 0405146v2`.

[Kha79] L. Khachiyan. "A polynomial algorithm in linear programming". In: *Dokl. Akad. Nauk SSSR* 244.5 (1979), pp. 1093–1096.

[Kim13] S. Kimmel. "Quantum Adversary (Upper) Bound". In: *Chicago Journal of Theoretical Computer Science* 4 (2013). arXiv: `1101.0797v5`.

[Kit95] A. Kitaev. "Quantum measurements and the Abelian stabiliser problem". 1995. arXiv: `quant-ph/9511026`.

[KKMO04] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. "Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?" In: *Proceedings of 45th IEEE FOCS*. 2004, pp. 146–154.

[KNP05] P. Koiran, V. Nesme, and N. Portier. "A Quantum Lower Bound for the Query Complexity of Simon's Problem". In: *Proceedings of the 32nd ICALP*. Vol. 3580. Lecture Notes in Computer Science. 2005, pp. 1287–1298.

[KNP07] P. Koiran, V. Nesme, and N. Portier. “The quantum query complexity of the Abelian hidden subgroup problem”. In: *Theoretical Computer Science* 380 (2007), pp. 115–126.

[KP17] I. Kerenidis and A. Prakash. “Quantum Recommendation Systems”. In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 2017, pp. 49:1–49:21. arXiv: `1603.08675v3`.

[KY14] C. Koufogiannakis and N. Young. “A Nearly Linear-Time PTAS for Explicit Fractional Packing and Covering Linear Programs”. In: *Algorithmica* 70.4 (2014), pp. 648–674.

[Las01] J. Lasserre. “Global Optimization with Polynomials and the Problem of Moments”. In: *SIAM Journal on Optimization* 11.3 (2001), pp. 796–817.

[LC16] G. Low and I. Chuang. “Hamiltonian Simulation by Qubitization”. 2016. arXiv: `1610.06546v3`.

[LC17] G. Low and I. Chuang. “Hamiltonian Simulation by Uniform Spectral Amplification”. 2017. arXiv: `1707.05391v1`.

[LN93] M. Luby and N. Nisan. “A Parallel Approximation Algorithm for Positive Linear Programming”. In: *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*. 1993, pp. 448–457.

[Low19] G. Low. “Hamiltonian simulation with nearly optimal dependence on spectral norm”. In: *Proceedings of the 51th ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 491–502. arXiv: `1807.03967v1`.

[LRS15] J. Lee, P. Raghavendra, and D. Steurer. “Lower Bounds on the Size of Semidefinite Programming Relaxations”. In: *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*. 2015, pp. 567–576. arXiv: `1411.6317v1`.

[LSV18] Y. Lee, A. Sidford, and S. Vempala. “Efficient Convex Optimization with Membership Oracles”. In: *Proceedings of the 31st Conference On Learning Theory (COLT)*. 2018, pp. 1292–1294. arXiv: `1706.07357`.

[LSW15] Y. Lee, A. Sidford, and S. Wong. “A faster cutting plane method and its implications for combinatorial and convex optimization”. In: *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2015, pp. 1049–1065. arXiv: `1508.04874`.

[LSWW14] Y. Li, X. Sun, C. Wang, and D. Woodruff. “On the Communication Complexity of Linear Algebraic Problems in the Message Passing Model”. In: *Distributed Computing*. 2014, pp. 499–513.

[Lux22] R. Luxemburg. *Die russische Revolution: eine kritische Würdigung*. Ed. by P. Levi. Berlin Verl. Gesellschaft und Erziehung, 1922. In German.

[Man80] Y. Manin. “Vychislimoe i nevychislimoe (Computable and Noncomputable)”. In: *Soviet Radio* (1980), pp. 13–15. In Russian.

[Man99] Y. Manin. “Classical computing, quantum computing, and Shor’s factoring algorithm”. 1999. arXiv: `quant-ph/9903008v1`.

[Mon15] A. Montanaro. “Quantum speedup of Monte Carlo methods”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2181 (2015). arXiv: `1504.06987v3`.

[Moo65] G. Moore. “Cramming more components onto integrated circuits”. In: *Electronics Magazine* 38.8 (1965).

[NC00] M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000.

[Neu47] J. von Neumann. *On a Maximization Problem*. 1947. (manuscript).

[NN94] Y. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming*. Vol. 13. SIAM Studies in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), 1994.

[Par00] Pablo A. Parrilo. “Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization”. PhD thesis. California Institute of Technology, 2000.

[Pla00a] M. Planck. *On an Improvement of Wien’s Equation for the Spectrum*. 1900. Read in German at a meeting of the Deutsche Physikalische Gesellschaft on 19 October 1900.

[Pla00b] M. Planck. *On the Theory of the Energy Distribution Law of the Normal Spectrum*. 1900. Read in German at a meeting of the Deutsche Physikalische Gesellschaft on 14 December 1900.

[RC66] T. Rivlin and E. Cheney. “A Comparison of Uniform Approximations on an Interval and a Finite Subset Thereof”. In: *SIAM Journal on Numerical Analysis* 3.2 (1966), pp. 311–320.

[Rei11] B. Reichardt. “Reflections for quantum query algorithms”. In: *Proceedings of 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2011, pp. 560–569. arXiv: `1005.1601v1`.

[Ren16] J. Renegar. ““Efficient” Subgradient Methods for General Convex Optimization”. In: *SIAM Journal on Optimization* 26.4 (2016), pp. 2649–2676.

[Ren19] J. Renegar. “Accelerated first-order methods for hyperbolic programming”. In: *Mathematical Programming* 173.1 (2019), pp. 1–35.

[Sch26] E. Schrödinger. “Quantisierung als Eigenwertproblem”. In: *Annalen der Physik* 384.6 (1926), pp. 489–527.

[Sho97] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. arXiv: `quant-ph/9508027v2`. Earlier version in FOCS’94.

[Sil80] S. Silvey. *Optimal Design: an introduction to the theory for parameter estimation*. Chapman and Hall Ltd, 1980.

[Sim97] D. Simon. “On the Power of Quantum Computation”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1474–1483. Earlier version in FOCS’94.

[ŠS05] R. Špalek and M. Szegedy. “All Quantum Adversary Methods are Equivalent”. In: *Proceedings of 32nd ICALP*. Vol. 3580. Lecture Notes in Computer Science. 2005, pp. 1299–1311. arXiv: `quant-ph/0409116v3`.

[SWYZ19] X. Sun, D. Woodruff, G. Yang, and J. Zhang. “Querying a Matrix Through Matrix-Vector Products”. In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). 2019, pp. 94:1–94:16.

[TRW05] K. Tsuda, G. Rätsch, and M. Warmuth. “Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection”. In: *Journal of Machine Learning Research* 6 (2005), pp. 995–1018. Earlier version in NIPS’04.

[Tsi87] B. Tsirel’son. “Quantum analogues of the Bell inequalities. The case of two spatially separated domains”. In: *Journal of Soviet Mathematics* 36.4 (1987), pp. 557–570.

[Tur36] A. Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. In: *Proceedings of the London Mathematical Society*. Vol. 2. 42. 1936, pp. 230–265.

[Wat18] J. Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018.

[WK12] M. Warmuth and D. Kuzmin. “Online variance minimization”. In: *Machine Learning* 87.1 (2012), pp. 1–32. Earlier version in COLT’06.

[Wol02] R. de Wolf. “Quantum communication and complexity”. In: *Theoretical Computer Science* 287.1 (2002), pp. 337–353.

[Wol19] R. de Wolf. “Quantum Computing: Lecture Notes”. 2019. arXiv: `1907.09415v1`.

[Wu17] X. Wu. *Personal communication*. Email. 2017.

[Yao75] A. Yao. “On Computing the Minima of Quadratic Forms (Preliminary Report)”. In: *Proceedings of the 7th ACM Symposium on the Theory of Computing (STOC)*. 1975, pp. 23–26.

Abstract

Every day we need to make many decisions: what we should have for dinner, what to vote for, or whether to kiss that special person for the first time. There are also many decisions that need to be made at a larger scale, for example, groups, governments, and corporations have to make decisions in order to use their resources wisely. Every decision asks us to consider many possibilities and to try to get the *optimal* outcome (or more often than not, to simply avoid disaster). Making these decisions is further complicated by the constraints that are put on us. Even though people have become quite good at making decisions, the increasing complexity of the world around us means that "going with our guts" is often an ill-advised optimization strategy.

The field of mathematics allows us to formulate an abstract version of some of the more complex problems we encounter, and allows us to quantify what the objectives and constraints are. For some problems mathematics even gives us a simple closed-form solution, e.g. many of us have learned in school how find the extreme points on a parabola. However, for many problems there is no such simple closed-form solution, the best we can do is to use a method, or *algorithm*, to construct a solution. This puts us in the realm of *computer science*.

Computer science considers how to solve problems systematically, and more importantly how many steps the decision-making process has to take. Most optimization problems can be solved by simply listing all possible combinations of decisions and then searching for the best option, but as the number of decisions increases, so does the number of combinations. In fact, for many situations the number of possible combinations increases *exponentially*, making the process of writing down all the options impractical. Instead we often look for a smarter, more *efficient* algorithm, one that scales better with the problem size.

It is not always easy (or even possible) to find an efficient algorithm. Consider the problem of finding the lowest place in the Swiss Alps. We may walk down into what seems the deepest valley, only to learn later that a farmer three valleys over has dug a really deep well. One of the problems in such a scenario is that we may find a *local minimum*, but due to a lack of structure in the problem we have no clear way of finding a *global minimum*. Some problems do have more structure, an important class of these is the class of *convex optimization problems*. Convex optimization problems have the useful property that local minima are also global minima. For our example about the Alps this means that, as long as we keep going down, we will always reach the lowest point. However, as anyone who ever went snowboarding can tell you, how you go down

matters a lot for how fast you will get there.

Apart from considering the structure of the problem we are trying to solve we should also consider what steps we can take to solve it. We may want to use pen and paper, use our laptop, or use a supercomputer, all of which function differently. However, these methods are not inherently different, with enough time we could perform the same computation as a supercomputer with nothing more than a pen and a piece of paper. Although this would be significantly slower, the scaling with the problem size would stay the same since the basic steps we can use are the same, only slower. What basic steps we may perform is ultimately dictated by the laws of physics: we can only change the state of our piece of paper, or our computer, in ways that are allowed by physics. Specifically, our piece of paper and our computers both follow the laws of *classical physics*, the physics that we see around us every day.

During the 20th century our understanding of physics has changed and we now know that classical physics does not give a full description of the natural world. One of the places where classical physics breaks down is at the scale of atoms and elementary particles, at this small scale the universe is governed by the laws of *quantum physics*. Quantum physics predicts behavior that is not possible in classical physics. Particles might be in a *superposition* of many different classical states, only becoming a specific one of those states at random when interacting with our classical world via a measurement. Miraculously, and contrary to classical randomness, this quantum randomness does not stem from our inherent lack of knowledge about the particle, the different parts of the superposition are all present until we measure and can *interfere* with each other. A natural question to ask is whether these phenomena can be used to our benefit when performing computations. A computer that works according to the rules of quantum physics is called a *Quantum computer*, such devices have a wider range of basic operations that they can perform than classical computers do.

This thesis asks the following central question

Can quantum computers solve convex optimization problems faster?

In order to answer this question we consider a few different types of problem that we may encounter in convex optimization.

If we have some *black-box* procedure to evaluate how good a proposed solution to a convex optimization problem is, for example a way of estimating the height of a point in the Alps or a method of estimating our profits given a business strategy, then a classical computer would have to try a lot of small changes in order to find the change that improves our objective the most. However, we show that a quantum computer can find the optimal direction for improvement *exponentially faster*.

We then consider specific categories of convex optimization problems, mainly *linear programming* and *semidefinite programming*. These problem categories include many natural problems and have been widely used both for theoretical and practical purposes. For example, linear programming can be used for finding the optimal strategies of zero-sum games, coming up with a diet plan, or planning resources. Semidef-

inite programming is a generalization of linear programming and can additionally be useful for designing optimal experiments, approximately solving hard problems, or optimize processes involving quantum mechanics. Our quantum algorithms for these problems use a natural connection between quantum physics and semidefinite programming: the solutions to a semidefinite programming problem correspond to the states of a quantum system. We show that quantum computers can solve semidefinite programming problems and linear programming problems *quadratically* faster than a classical computer can. In fact, our algorithms take less time to solve the problem than that it would take to read through the whole input.

We also take a more negative view and discuss the limitations of quantum computers. We show that, even though quantum computers can find the optimal direction of improvement exponentially faster, there is no general exponential speedup for finding the optimal value. We also show that in many ways our methods for solving linear and semidefinite programming problems are optimal: faster methods would not be able to read enough of the input to still get a good solution.

To conclude, we showed that in a few different settings the answer to our central question is *yes*. The results in this dissertation show that quantum computers can solve some convex optimization problems faster, but there are also limits to this. There are still many open questions regarding convex optimization on quantum computers and we regard this topic as an important direction of future research.

Samenvatting

Iedere dag moeten we veel verschillende beslissingen nemen: wat te eten vanavond, waar op te stemmen, en of we die ene speciale persoon voor de eerste keer zullen zoenen. Ook op een grotere schaal moeten er veel beslissingen genomen worden; groepen, overheden, en bedrijven moeten bijvoorbeeld beslissingen nemen om hun middelen goed te benutten. Voor iedere beslissing moeten we rekening houden met veel verschillende mogelijkheden om zo een *optimale* uitkomst te bereiken (alhoewel we vaak al tevreden zijn als we simpelweg een ramp kunnen voorkomen). Het nemen van al deze beslissingen wordt alleen nog maar ingewikkelder als we ook rekening moeten houden met alle beperkingen die ons worden opgelegd. Hoewel mensen redelijk goed zijn in ingewikkelde beslissingen nemen, wordt de wereld om ons heen ook steeds ingewikkelder en is simpelweg dingen op gevoel doen vaak een slechte optimalisatie-strategie.

Gelukkig staat de wiskunde ons toe om een abstracte versie van ingewikkelde problemen te formuleren en zo duidelijk vast te leggen wat het doel en de beperkingen zijn. Voor sommige problemen krijgen we zelfs een oplossing in gesloten vorm, zo hebben veel van ons op school geleerd hoe we het extreme punt op een parabool kunnen vinden. Echter, er zijn ook problemen zonder zo'n simpele oplossing in gesloten vorm. Voor dit soort problemen kunnen we vaak alleen maar een *algoritme* vinden om het probleem op te lossen, de problemen vallen dan onder het vakgebied van *informatica*.

In de informatica vragen we ons af hoe we problemen op een systematische wijze kunnen oplossen. Vaak is het hierbij belangrijk hoeveel stappen nodig zijn voor het besluitvormingsproces. Zo zouden de meeste optimalisatieproblemen opgelost kunnen worden door alle mogelijke combinaties voor alle beslissingen uit te schrijven en dan de beste optie te kiezen, maar als het aantal te maken beslissingen groeit dan groeit ook het aantal combinaties dat we moeten uitwerken. Dit aantal combinaties zal vaak zelfs *exponentieel* groeien, waardoor deze strategie niet praktisch zal zijn. In dat geval zoeken we naar slimmere en efficiëntere algoritmes die beter schalen.

Het is niet altijd makkelijk (of zelf mogelijk) om een efficiënt algoritme te vinden. Neem bijvoorbeeld het probleem om het laagste punt in de Zwitserse Alpen te vinden. Je begint mogelijk op de top van een berg en loopt het dal in dat het diepste lijkt, maar later blijkt dat een boer een paar valleien verderop een hele diepe put heeft gegraven. Het probleem in dit geval is dat je mogelijk een *lokaal minimum* vind, maar door een gebrek aan structuur is het niet duidelijk hoe we een *globaal minimum* kunnen vinden. Sommige problemen hebben wel meer structuur. Een belangrijke klasse van zulke problemen zijn de *convexe optimalisatieproblemen*. Convexe optimalisatieproblemen hebben de fijne eigenschap dat lokale minima ook globale minima zijn. In het voorbeeld over de Alpen betekent dit dat, zolang we maar naar beneden blijven gaan, we altijd op het laagste punt aan zullen komen. Echter, zoals elke snowboarder je kan vertellen, speelt hoe je naar beneden gaat een grote rol bij hoe snel je uiteindelijk zal zijn.

Behalve naar de structuur van het probleem te kijken, dienen we er ook over te denken welke stappen we kunnen nemen om het op te lossen. Zo kunnen we een probleem met pen en papier oplossen, met onze laptop, of met een supercomputer. Allemaal verschillende methodes die anders zullen functioneren. Echter, deze methodes zijn niet inherent verschillend: als we genoeg tijd nemen kunnen we dezelfde berekeningen als een supercomputer uitvoeren met pen en papier. Hoewel dit significant trager zal zijn, zal de schaling met de probleemgrote wel hetzelfde zijn, we kunnen immers dezelfde basisstappen uitvoeren (alleen trager). Welke basisstappen we kunnen uitvoeren wordt uiteindelijk bepaald door de natuurwetten: we kunnen de staat van ons stuk papier (of onze computer) alleen maar veranderen op manieren die toegestaan worden door de natuurwetten. Zowel ons papier als onze computers opereren volgens de *klassieke natuurkunde*, de natuurkunde die het dagelijks leven om ons heen beschrijft.

In de 20e eeuw is onze blik op de natuurkunde veranderd, we weten nu dat de klassieke natuurkunde geen volledige beschrijving geeft van de wereld. De klassieke fysica werkt bijvoorbeeld niet op de kleinste schalen, de schaal van atomen en elementaire deeltjes. Op dit soort kleine schalen gedragen de deeltjes zich volgens de wetten van de quantumfysica, dit staat gedrag toe dat in onze klassieke dagelijkse wereld niet mogelijk is. Zo kunnen deeltjes zich in een *superpositie* van toestanden bevinden, een deeltje wordt willekeurig een van deze toestanden nadat er interactie met onze klassieke wereld heeft plaatsgevonden via een meting. Wonderbaarlijk genoeg is deze willekeur anders dan de klassieke willekeurigheid die voortkomt uit ons gebrek aan kennis over een deeltje: de verschillende delen van een superpositie zijn echt allemaal aanwezig totdat er gemeten word en kunnen met elkaar *interfereren*. Een natuurlijke vraag is dan ook of deze quantum verschijnselen ons kunnen helpen bij het uitvoeren van berekeningen. Een computer die zich gedraagt volgens de regels van de quantummechanica heet een *quantumcomputer* en kan een grotere set aan basisoperaties uitvoeren dan een klassieke computer.

In dit proefschrift staat de volgende vraag centraal:

Kunnen quantumcomputers convexe optimalisatieproblemen sneller oplossen?

Om deze vraag te beantwoorden kijken we naar een verschillende soorten problemen die voorkomen in convexe optimalisatie.

Zo kijken we bijvoorbeeld naar optimalisatiemethodes voor wanneer we kunnen bepalen hoe goed een mogelijke oplossing voor een probleem is. We zijn specifiek geïnteresseerd in het geval dat we hiertoe toegang hebben in een “black-box manner”, met andere worden, we weten niet hoe deze evaluatie achter de schermen werkt. Zo’n manier van evalueren kan bijvoorbeeld een manier zijn om de hoogte van een punt in de Alpen te schatten of een manier om te bepalen hoe goed een handelsstrategie is. Gegeven een mogelijke oplossing zou een klassiek algoritme alle verschillende veranderingen moeten uitproberen om te zien welke aanpassing de grootste verbetering oplevert. We tonen echter aan dat je met een quantumcomputer exponentieel sneller een optimale aanpassing kan vinden.

We kijken daarna naar bepaalde klassen aan convexe optimalisatieproblemen, specifiek kijken we naar *lineair programmeren* en *semidefiniet programmeren*. Deze categorieën bevatten veel natuurlijke problemen en zijn breed toepasbaar voor zowel theoretische als praktische doeleinden. Bijvoorbeeld, lineair programmeren kan gebruikt worden om de optimale strategieën in nulsomspelen uit te rekenen, om een dieetplan op te stellen, of om middelen slim te verdelen. Semidefiniet programmeren is een veralgemenisering van lineair programmeren en kan ook nog gebruikt worden om bijvoorbeeld optimale experimenten te ontwerpen, moeilijke problemen te benaderen, en om processen waar quantummechanica een rol speelt te optimaliseren. Onze quantumalgoritmen voor deze problemen gebruiken een natuurlijke connectie tussen quantumfysica en semidefiniet programmeren: de oplossingen van semidefiniete programma's komen overeen met de toestanden van een quantumsysteem. We laten zien dat quantumcomputers semidifiniete programma's en lineaire programma's *kwadratisch* sneller kunnen oplossen dan klassieke computers dat kunnen. Onze algoritmen lossen het probleem zelfs sneller op dan de tijd die een klassieke computer nodig zou hebben om door de hele invoer heen te lezen.

We kijken echter ook naar de beperkingen van quantumcomputers. Zo laten we zien dat, ondanks dat quantumcomputers exponentieel sneller de optimale richting voor verandering kunnen bepalen, er geen algemene exponentiële verbetering is voor het vinden van een optimale oplossing. Ook laten we zien dat op veel verschillende manieren onze methodes voor het oplossen van semidefiniete en lineaire programma's niet verder te versnellen zijn: snellere methodes zouden te weinig van de invoer moeten bekijken en daardoor geen goede oplossing meer kunnen vinden.

Samengevat, we laten zien dat er meerdere situaties zijn waarin het antwoord op onze centrale vraag *ja* is. De resultaten in dit proefschrift laten zien dat quantumcomputers sommige convexe optimalisatieproblemen sneller kunnen oplossen, maar dit is wel beperkt. Er zijn nog vele open vragen over convexe optimalisatie op quantumcomputers en we zien dit dan ook als een belangrijk onderwerp voor vervolgonderzoek.

Titles in the ILLC Dissertation Series:

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic

ILLC DS-2009-05: **Andreas Witzel**
Knowledge and Games: Theory and Implementation

ILLC DS-2009-06: **Chantal Bax**
Subjectivity after Wittgenstein. Wittgenstein's embodied and embedded subject and the debate about the death of man.

ILLC DS-2009-07: **Kata Balogh**
Theme with Variations. A Context-based Analysis of Focus

ILLC DS-2009-08: **Tomohiro Hoshi**
Epistemic Dynamics and Protocol Information

ILLC DS-2009-09: **Olivia Ladinig**
Temporal expectations and their violations

ILLC DS-2009-10: **Tikitu de Jager**
"Now that you mention it, I wonder...": Awareness, Attention, Assumption

ILLC DS-2009-11: **Michael Franke**
Signal to Act: Game Theory in Pragmatics

ILLC DS-2009-12: **Joel Uckelman**
More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains

ILLC DS-2009-13: **Stefan Bold**
Cardinals as Ultrapowers. A Canonical Measure Analysis under the Axiom of Determinacy.

ILLC DS-2010-01: **Reut Tsarfaty**
Relational-Realizational Parsing

ILLC DS-2010-02: **Jonathan Zvesper**
Playing with Information

ILLC DS-2010-03: **Cédric Dégremont**
The Temporal Mind. Observations on the logic of belief change in interactive systems

ILLC DS-2010-04: **Daisuke Ikegami**
Games in Set Theory and Logic

ILLC DS-2010-05: **Jarmo Kontinen**
Coherence and Complexity in Fragments of Dependence Logic

ILLC DS-2010-06: **Yanjing Wang**
Epistemic Modelling and Protocol Dynamics

ILLC DS-2010-07: **Marc Staudacher**
Use theories of meaning between conventions and social norms

ILLC DS-2010-08: **Amélie Gheerbrant**
Fixed-Point Logics on Trees

ILLC DS-2010-09: **Gaëlle Fontaine**
Modal Fixpoint Logic: Some Model Theoretic Questions

ILLC DS-2010-10: **Jacob Vosmaer**
Logic, Algebra and Topology. Investigations into canonical extensions, duality theory and point-free topology.

ILLC DS-2010-11: **Nina Gierasimczuk**
Knowing One's Limits. Logical Analysis of Inductive Inference

ILLC DS-2010-12: **Martin Mose Bentzen**
Stit, Iit, and Deontic Logic for Action Types

ILLC DS-2011-01: **Wouter M. Koolen**
Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice

ILLC DS-2011-02: **Fernando Raymundo Velazquez-Quesada**
Small steps in dynamics of information

ILLC DS-2011-03: **Marijn Koolen**
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

ILLC DS-2011-04: **Junte Zhang**
System Evaluation of Archival Description and Access

ILLC DS-2011-05: **Lauri Keskinen**
Characterizing All Models in Infinite Cardinalities

ILLC DS-2011-06: **Rianne Kaptein**
Effective Focused Retrieval by Exploiting Query Context and Document Structure

ILLC DS-2011-07: **Jop Briët**
Grothendieck Inequalities, Nonlocal Games and Optimization

ILLC DS-2011-08: **Stefan Minica**
Dynamic Logic of Questions

ILLC DS-2011-09: **Raul Andres Leal**
Modalities Through the Looking Glass: A study on coalgebraic modal logic and their applications

ILLC DS-2011-10: **Lena Kurzen**
Complexity in Interaction

ILLC DS-2011-11: **Gideon Borensztajn**
The neural basis of structure in language

ILLC DS-2012-01: **Federico Sangati**
Decomposing and Regenerating Syntactic Trees

ILLC DS-2012-02: **Markos Mylonakis**
Learning the Latent Structure of Translation

ILLC DS-2012-03: **Edgar José Andrade Lotero**
Models of Language: Towards a practice-based account of information in natural language

ILLC DS-2012-04: **Yurii Khomskii**
Regularity Properties and Definability in the Real Number Continuum: idealized forcing, polarized partitions, Hausdorff gaps and mad families in the projective hierarchy.

ILLC DS-2012-05: **David García Soriano**
Query-Efficient Computation in Property Testing and Learning Theory

ILLC DS-2012-06: **Dimitris Gakis**
Contextual Metaphilosophy - The Case of Wittgenstein

ILLC DS-2012-07: **Pietro Galliani**
The Dynamics of Imperfect Information

ILLC DS-2012-08: **Umberto Grandi**
Binary Aggregation with Integrity Constraints

ILLC DS-2012-09: **Wesley Halcrow Holliday**
Knowing What Follows: Epistemic Closure and Epistemic Logic

ILLC DS-2012-10: **Jeremy Meyers**
Locations, Bodies, and Sets: A model theoretic investigation into nominalistic mereologies

ILLC DS-2012-11: **Floor Sietsma**
Logics of Communication and Knowledge

ILLC DS-2012-12: **Joris Dormans**
Engineering emergence: applied theory for game design

ILLC DS-2013-01: **Simon Pauw**
Size Matters: Grounding Quantifiers in Spatial Perception

ILLC DS-2013-02: **Virginie Fiutek**
Playing with Knowledge and Belief

ILLC DS-2013-03: **Giannicola Scarpa**
Quantum entanglement in non-local games, graph parameters and zero-error information theory

ILLC DS-2014-01: **Machiel Keestra**
Sculpting the Space of Actions. Explaining Human Action by Integrating Intentions and Mechanisms

ILLC DS-2014-02: **Thomas Icard**
The Algorithmic Mind: A Study of Inference in Action

ILLC DS-2014-03: **Harald A. Bastiaanse**
Very, Many, Small, Penguins

ILLC DS-2014-04: **Ben Rodenhäuser**
A Matter of Trust: Dynamic Attitudes in Epistemic Logic

ILLC DS-2015-01: **María Inés Crespo**
Affecting Meaning. Subjectivity and evaluativity in gradable adjectives.

ILLC DS-2015-02: **Mathias Winther Madsen**
The Kid, the Clerk, and the Gambler - Critical Studies in Statistics and Cognitive Science

ILLC DS-2015-03: **Shengyang Zhong**
Orthogonality and Quantum Geometry: Towards a Relational Reconstruction of Quantum Theory

ILLC DS-2015-04: **Sumit Sourabh**
Correspondence and Canonicity in Non-Classical Logic

ILLC DS-2015-05: **Facundo Carreiro**
Fragments of Fixpoint Logics: Automata and Expressiveness

ILLC DS-2016-01: **Ivano A. Ciardelli**
Questions in Logic

ILLC DS-2016-02: **Zoé Christoff**
Dynamic Logics of Networks: Information Flow and the Spread of Opinion

ILLC DS-2016-03: **Fleur Leonie Bouwer**
What do we need to hear a beat? The influence of attention, musical abilities, and accents on the perception of metrical rhythm

ILLC DS-2016-04: **Johannes Marti**
Interpreting Linguistic Behavior with Possible World Models

ILLC DS-2016-05: **Phong Lê**
Learning Vector Representations for Sentences - The Recursive Deep Learning Approach

ILLC DS-2016-06: **Gideon Maillette de Buy Wenniger**
Aligning the Foundations of Hierarchical Statistical Machine Translation

ILLC DS-2016-07: **Andreas van Cranenburgh**
Rich Statistical Parsing and Literary Language

ILLC DS-2016-08: **Florian Speelman**
Position-based Quantum Cryptography and Catalytic Computation

ILLC DS-2016-09: **Teresa Piovesan**
Quantum entanglement: insights via graph parameters and conic optimization

ILLC DS-2016-10: **Paula Henk**
Nonstandard Provability for Peano Arithmetic. A Modal Perspective

ILLC DS-2017-01: **Paolo Galeazzi**
Play Without Regret

ILLC DS-2017-02: **Riccardo Pinosio**
The Logic of Kant's Temporal Continuum

ILLC DS-2017-03: **Matthijs Westera**
Exhaustivity and intonation: a unified theory

ILLC DS-2017-04: **Giovanni Cinà**
Categories for the working modal logician

ILLC DS-2017-05: **Shane Noah Steinert-Threlkeld**
Communication and Computation: New Questions About Compositionality

ILLC DS-2017-06: **Peter Hawke**
The Problem of Epistemic Relevance

ILLC DS-2017-07: **Aybüke Özgün**
Evidence in Epistemic Logic: A Topological Perspective

ILLC DS-2017-08: **Raquel Garrido Alhama**
Computational Modelling of Artificial Language Learning: Retention, Recognition & Recurrence

ILLC DS-2017-09: **Milo Stanojevi**
Permutation Forests for Modeling Word Order in Machine Translation

ILLC DS-2018-01: **Berit Janssen**
Retained or Lost in Transmission? Analyzing and Predicting Stability in Dutch Folk Songs

ILLC DS-2018-02: **Hugo Huurdeman**
Supporting the Complex Dynamics of the Information Seeking Process

ILLC DS-2018-03: **Corina Koolen**
Reading beyond the female: The relationship between perception of author gender and literary quality

ILLC DS-2018-04: **Jelle Bruineberg**
Anticipating Affordances: Intentionality in self-organizing brain-body-environment systems

ILLC DS-2018-05: **Joachim Daiber**
Typologically Robust Statistical Machine Translation: Understanding and Exploiting Differences and Similarities Between Languages in Machine Translation

ILLC DS-2018-06: **Thomas Brochhagen**
Signaling under Uncertainty

ILLC DS-2018-07: **Julian Schlöder**
Assertion and Rejection

ILLC DS-2018-08: **Srinivasan Arunachalam**
Quantum Algorithms and Learning Theory

ILLC DS-2018-09: **Hugo de Holanda Cunha Nobrega**
Games for functions: Baire classes, Weihrauch degrees, transfinite computations, and ranks

ILLC DS-2018-10: **Chenwei Shi**
Reason to Believe

ILLC DS-2018-11: **Malvin Gattinger**
New Directions in Model Checking Dynamic Epistemic Logic

ILLC DS-2018-12: **Julia Ilin**
Filtration Revisited: Lattices of Stable Non-Classical Logics

ILLC DS-2018-13: **Jeroen Zuiddam**
Algebraic complexity, asymptotic spectra and entanglement polytopes

ILLC DS-2019-01: **Carlos Vaquero**
What Makes A Performer Unique? Idiosyncrasies and commonalities in expressive music performance

ILLC DS-2019-02: **Jort Bergfeld**
Quantum logics for expressing and proving the correctness of quantum programs

ILLC DS-2019-03: **András Gilyén**
Quantum Singular Value Transformation & Its Algorithmic Applications

ILLC DS-2019-04: **Lorenzo Galeotti**
The theory of the generalised real numbers and other topics in logic

ILLC DS-2019-05: **Nadine Theiler**
Taking a unified perspective: Resolutions and highlighting in the semantics of attitudes and particles

ILLC DS-2019-06: **Peter T.S. van der Gulik**
Considerations in Evolutionary Biochemistry

ILLC DS-2019-07: **Frederik Möllerström Lauridsen**
Cuts and Completions: Algebraic aspects of structural proof theory

ILLC DS-2020-01: **Mostafa Dehghani**
Learning with Imperfect Supervision for Language Understanding

ILLC DS-2020-02: **Koen Groenland**
Quantum protocols for few-qubit devices

ILLC DS-2020-03: **Jouke Witteveen**
Parameterized Analysis of Complexity

ILLC DS-2020-04: **Joran van Apeldoorn**
A Quantum View on Convex Optimization