Quantum SDP-Solvers: Better upper and lower bounds

Joran van Apeldoorn[1], András Gilyén[1], Sander Gribling[1], and Ronald de Wolf[2]

[1]QuSoft, CWI, Amsterdam, the Netherlands. {apeldoor,gilyen,gribling}@cwi.nl

[2]QuSoft, CWI and University of Amsterdam, the Netherlands. rdewolf@cwi.nl

Brandão and Svore [BS17] recently gave quantum algorithms for approximately solving semidefinite programs, which in some regimes are faster than the best-possible classical algorithms in terms of the dimension n of the problem and the number m of constraints, but worse in terms of various other parameters. In this paper we improve their algorithms in several ways, getting better dependence on those other parameters. To this end we develop new techniques for quantum algorithms, for instance a general way to efficiently implement smooth functions of sparse Hamiltonians, and a generalized minimum-finding procedure.

We also show limits on this approach to quantum SDP-solvers, for instance for combinatorial optimization problems that have a lot of symmetry. Finally, we prove some general lower bounds showing that in the worst case, the complexity of every quantum LP-solver (and hence also SDP-solver) has to scale linearly with mn when $m \approx n$, which is the same as classical.

Contents

1 Introduction **3**
- 1.1 Semidefinite programs . . . 3
- 1.2 Classical solvers for LPs and SDPs . . . 4
- 1.3 Quantum SDP-solvers: the Brandão-Svore algorithm . . . 5
- 1.4 Our results . . . 6
  - 1.4.1 Improved quantum SDP-solver . . . 6
  - 1.4.2 Tools that may be of more general interest . . . 7
  - 1.4.3 Lower bounds . . . 8

2 An improved quantum SDP-solver **10**
- 2.1 The Arora-Kale framework for solving SDPs . . . 11
- 2.2 Approximating the expectation value $\mathrm{Tr}(A\rho)$ using a quantum algorithm . . . 16
  - 2.2.1 General approach . . . 16
  - 2.2.2 The special case of diagonal matrices – for LP-solving . . . 18
  - 2.2.3 General case – for SDP-solving . . . 19
- 2.3 An efficient 2-sparse oracle . . . 22
- 2.4 Total runtime . . . 26

3 Downside of this method: general oracles are restrictive **28**
- 3.1 Sparse oracles are restrictive . . . 29
- 3.2 General width-bounds are restrictive for certain SDPs . . . 30

4 Lower bounds on the quantum query complexity **35**

5 Conclusion **38**

A Classical estimation of the expectation value $\mathrm{Tr}(A\rho)$ **42**

B Implementing smooth functions of Hamiltonians **46**
- B.1 Implementation of smooth functions of Hamiltonians: general results . . . 48
- B.2 Applications of smooth functions of Hamiltonians . . . 55

C Generalized minimum-finding algorithm **58**

D Sparse matrix summation **65**
- D.1 A lower bound . . . 65
- D.2 An upper bound . . . 66

E Equivalence of R**,** r**, and** ε^{-1} **66**

1 Introduction

1.1 Semidefinite programs

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 the better-known *linear* programs (LPs), but (like LPs) they are still efficiently solvable. Thanks to their stronger expressive power, SDPs can sometimes give better approximation algorithms than LPs.

The basic form of an SDP is the following:

$$\begin{aligned} \max \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \le b_j \quad \text{for all } j \in [m], \\ & X \succeq 0, \end{aligned} \tag{1}$$

where $[m] := \{1, \ldots, m\}$. The input to the problem consists of Hermitian $n \times n$ matrices $C, A_1, \ldots, A_m$ and reals $b_1, \ldots, b_m$. For normalization purposes we assume $\|C\|, \|A_j\| \le 1$. 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. LPs correspond to the case where all matrices are diagonal.

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

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

using the fact that $(1 - v_i v_j)/2 = 1$ if v_i and v_j are different signs, and $(1 - v_i v_j)/2 = 0$ 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 $v_1, \ldots, v_n$, so $X_{ij} = v_i^T v_j$. The resulting SDP is

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

where E_{jj} is the $n \times n$ matrix that has a 1 at the (j, j)-entry, and 0s elsewhere.

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 MAXCUT(G) (which is optimal under the Unique Games Conjecture [KKMO07]). This SDP can be massaged into the form of (1) by replacing the equality $\mathrm{Tr}(E_{jj} X) = 1$ by inequality $\mathrm{Tr}(E_{jj} X) \le 1$ (so $m = n$) and letting C be a properly normalized version of the Laplacian of G.

1.2 Classical solvers for LPs and SDPs

Ever since Dantzig's development of the simplex algorithm for solving LPs in the 1940s [Dan51], 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] approximate the optimal value OPT of such an SDP up to additive error ε, with complexity

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

where $\omega \in [2, 2.373)$ is the (still unknown) optimal exponent for matrix multiplication; s is the *sparsity*: the maximal number of non-zero entries per row of the input matrices; and R is an upper bound on the trace of an optimal X.[1] The assumption here is that the rows and columns of the matrices of SDP (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.[2] In Section 2.1 we will describe their framework in more detail, but in order to describe our result we will start with an overly simplified sketch here. The algorithm goes back and forth between candidate solutions to the primal SDP and to the corresponding *dual* SDP, whose variables are non-negative reals $y_1, \ldots, y_m$:

$$\begin{aligned} \min \quad & b^T y \\ \text{s.t.} \quad & \sum_{j=1}^{m} y_j A_j - C \succeq 0, \\ & y \geq 0. \end{aligned} \tag{2}$$

Under assumptions that will be satisfied everywhere in this paper, strong duality applies: the primal SDP (1) and dual SDP (2) will have the same optimal value OPT. The algorithm does a binary search for OPT by trying different guesses α for it. Suppose we have fixed some α, and want to find out whether α is bigger or smaller than OPT. Start with some candidate solution $X^{(1)}$ for the primal, for example a multiple of the identity matrix ($X^{(1)}$ has to be psd but need not be a *feasible* solution to the primal). This $X^{(1)}$ induces the following polytope:

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

This polytope can be thought of as a relaxation of the feasible region of the dual SDP with the extra constraint that OPT $\leq \alpha$: instead of requiring that $\sum_j y_j A_j - C$ is psd, we merely

[1] See Lee, Sidford, and Wong [LSW15, Section 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, Theorem 45 of arXiv version 2] that in their setting $r \leq mR$, and hence r is absorbed in the $\log^{O(1)}(mnR/\varepsilon)$ factor.

[2] 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].

require that its inner product with the particular psd matrix $X^{(1)}$ is not too negative. The algorithm then calls an "oracle" that provides a $y^{(1)} \in \mathcal{P}_\varepsilon(X^{(1)})$, or outputs "fail" if $\mathcal{P}_0(X^{(1)})$ is empty (how to efficiently implement such an oracle depends on the application). In the "fail" case we know there is no dual-feasible y with objective value $\leq \alpha$, so we can increase our guess α for OPT, and restart. In case the oracle produced a $y^{(1)}$, this is used to define a Hermitian matrix $H^{(1)}$ and a new candidate solution $X^{(2)}$ for the primal, which is proportional to $e^{-H^{(1)}}$. Then the oracle for the polytope $\mathcal{P}_\varepsilon(X^{(2)})$ induced by this $X^{(2)}$ is called to produce a candidate $y^{(2)} \in \mathcal{P}_\varepsilon(X^{(2)})$ for the dual (or "fail"), this is used to define $H^{(2)}$ and $X^{(3)}$ proportional to $e^{-H^{(2)}}$, and so on.

Surprisingly, the average of the dual candidates $y^{(1)}, y^{(2)}, \ldots$ converges to a nearly-dual-feasible solution. Let R be an upper bound on the trace of an optimal X of the primal, r be an upper bound on the sum of entries of an optimal y for the dual, and w^* be the "width" of the oracle for a certain SDP: the maximum of $\left\|\sum_{j=1}^m y_j A_j - C\right\|$ over all psd matrices X and all vectors y that the oracle may output for the corresponding polytope $\mathcal{P}_\varepsilon(X)$. In general we will not know the width of an oracle exactly, but only an upper bound $w \geq w^*$, that may depend on the SDP; this is, however, enough for the Arora-Kale framework. In Section 2.1 we will show that without loss of generality we can assume the oracle returns a y such that $\|y\|_1 \leq r$. Because we assumed $\|A_j\|, \|C\| \leq 1$, we have $w^* \leq r + 1$ as an easy width-bound. General properties of the multiplicative weights update method guarantee that after $T = \widetilde{\mathcal{O}}(w^2 R^2/\varepsilon^2)$ iterations[3], if no oracle call yielded "fail", then the vector $\frac{1}{T}\sum_{t=1}^T y^{(t)}$ is close to dual-feasible and satisfies $b^T y \leq \alpha$. This vector can then be turned into a dual-feasible solution by tweaking its first coordinate, certifying that OPT $\leq \alpha + \varepsilon$, and we can decrease our guess α for OPT accordingly.

The framework of Arora and Kale is really a meta-algorithm, because it does not specify how to implement the oracle. They themselves provide oracles that are optimized for special cases, which allows them to give a very low width-bound for 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 the matrix $X^{(t)}$ in each iteration: the algorithm can still be made to work if instead of providing the oracle with $X^{(t)}$, we feed it good estimates of $\mathrm{Tr}(A_j X^{(t)})$ and $\mathrm{Tr}(C X^{(t)})$. Arora and Kale do not describe oracles for general SDPs, but as we show at the end of Section 2.4 (using Appendix A to estimate $\mathrm{Tr}(A_j X^{(t)})$ and $\mathrm{Tr}(C X^{(t)})$), one can get a general classical SDP-solver in their framework with complexity

$$\widetilde{\mathcal{O}}\left(nms\left(\frac{Rr}{\varepsilon}\right)^4 + ns\left(\frac{Rr}{\varepsilon}\right)^7\right). \tag{3}$$

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. It should be noted that for many specific cases, Arora and Kale get significantly better upper bounds than (3) by designing oracles that are specifically optimized for those cases.

1.3 Quantum SDP-solvers: the Brandão-Svore algorithm

Given the speed-ups that *quantum* computers give over classical computers for various problems [Sho97, Gro96, DHHM06, Amb07, HHL09], it is natural to ask whether quantum computers

[3]The $\widetilde{\mathcal{O}}(\cdot)$ notation hides polylogarithmic factors in all parameters.

can solve LPs and SDPs more efficiently as well. Very little was known about this, until recently when Brandão and Svore [BS17] discovered quantum algorithms that significantly outperform classical SDP-solvers in certain regimes. Because of the general importance of quickly solving LPs and SDPs, and the limited number of quantum algorithms that have been found so far, this is a very interesting development.

The key idea of the Brandão-Svore algorithm is to take the Arora-Kale approach and to replace two of its steps by more efficient quantum subroutines. First, given a vector $y^{(t-1)}$, it turns out one can use "Gibbs sampling" to prepare the new primal candidate $X^{(t)} \propto e^{-H^{(t-1)}}$ *as a* $\log(n)$*-qubit quantum state* $\rho^{(t)} := X^{(t)}/\mathrm{Tr}(X^{(t)})$ in much less time than needed to compute $X^{(t)}$ as an $n \times n$ matrix. Second, one can efficiently implement the oracle for $\mathcal{P}_\varepsilon(X^{(t)})$ based on a number of copies of $\rho^{(t)}$, using those copies to estimate $\mathrm{Tr}(A_j\rho^{(t)})$ and $\mathrm{Tr}(A_jX^{(t)})$ when needed (note that $\mathrm{Tr}(A\rho)$ is the expectation value of operator A for the quantum state ρ). This is based on something called "Jaynes's principle." The resulting oracle is weaker than what is used classically, in the sense that it outputs a sample $j \sim y_j/\|y\|_1$ rather than the whole vector y. However, such sampling still suffices to make the algorithm work (it also means we can assume the vector $y^{(t)}$ to be quite sparse).

Using these ideas, Brandão and Svore obtain a quantum SDP-solver of complexity

$$\widetilde{\mathcal{O}}(\sqrt{mn}s^2R^{32}/\delta^{18}),$$

with *multiplicative* error $1 \pm \delta$ for the special case where $b_j \geq 1$ for all $j \in [m]$, and $\mathrm{OPT} \geq 1$ (the latter assumption allows them to convert additive error ε to multiplicative error δ) [BS17, Corollary 5 in arXiv version 4]. They describe a reduction to transform a general SDP of the form (1) to this special case, but that reduction significantly worsens the dependence of the complexity on the parameters R, r, and δ.

Note that compared to the runtime (3) of our general instantiation of the original Arora-Kale framework, there are quadratic improvements in both m and n, corresponding to the two quantum modifications made to Arora-Kale. However, the dependence on R, r, s and $1/\varepsilon$ is much worse now than in (3). This quantum algorithm thus provides a speed-up only in regimes where $R, r, s, 1/\varepsilon$ are fairly small compared to mn (finding good examples of SDPs in such regimes is an open problem).

1.4 Our results

In this paper we present two sets of results: improvements to the Brandão-Svore algorithm, and better lower bounds for the complexity of quantum LP-solvers (and hence for quantum SDP-solvers as well).

1.4.1 Improved quantum SDP-solver

Our quantum SDP-solver, like the Brandão-Svore algorithm, works by quantizing some aspects of the Arora-Kale algorithm. However, the way we quantize is different and faster than theirs.

First, we give a more efficient procedure to estimate the quantities $\mathrm{Tr}(A_j\rho^{(t)})$ required by the oracle. Instead of first preparing some copies of Gibbs state $\rho^{(t)} \propto e^{-H^{(t-1)}}$ as a mixed state, we coherently prepare a purification of $\rho^{(t)}$, which can then be used to estimate $\mathrm{Tr}(A_j\rho^{(t)})$ more efficiently using amplitude-estimation techniques. Also, our purified Gibbs sampler has logarithmic dependence on the error, which is exponentially better than the Gibbs sampler of Poulin and Wocjan [PW09b] that Brandão and Svore invoke. Chowdhury and Somma [CS17] also gave a Gibbs sampler with logarithmic error-dependence, but assuming query access to the entries of $\sqrt{H}$ rather than H itself.

Second, we have a different implementation of the oracle, without using Gibbs sampling or Jaynes's principle (though, as mentioned above, we still use purified Gibbs sampling for approximating the $\mathrm{Tr}(A\rho)$ quantities). We observe that the vector $y^{(t)}$ can be made very sparse: *two* non-zero entries suffice.[4] We then show how we can efficiently find such a 2-sparse vector (rather than merely sampling from it) using two applications of a new generalization of the well-known quantum minimum-finding algorithm of Dürr and Høyer [DH96], which is based on Grover search [Gro96].

These modifications both simplify and speed up the quantum SDP-solver, resulting in complexity

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

The dependence on m, n, and s is the same as in Brandão-Svore, but our dependence on R, r, and $1/\varepsilon$ is substantially better. Note that each of the three parameters R, r, and $1/\varepsilon$ now occurs with the same 8th power in the complexity. This is no coincidence: as we show in Appendix E, these three parameters can all be traded for one another, in the sense that we can massage the SDP to make each one of them small at the expense of making the others proportionally bigger. These trade-offs suggest we should actually think of Rr/ε as *one* parameter of the primal-dual pair of SDPs, not three separate parameters. For the special case of LPs, we can improve the runtime to

$$\widetilde{\mathcal{O}}(\sqrt{mn}(Rr/\varepsilon)^5).$$

Like in Brandão-Svore, our quantum oracle produces very sparse vectors y, in our case even of sparsity 2. This means that after T iterations, the final ε-optimal dual-feasible vector (which is a slightly tweaked version of the average of the T y-vectors produced in the T iterations) has only $\mathcal{O}(T)$ non-zero entries. Such sparse vectors have some advantages, for example they take much less space to store than arbitrary $y \in \mathbb{R}^m$. In fact, to get a sublinear runtime in terms of m, this is necessary. However, this sparsity of the algorithm's output also points to a weakness of these methods: if *every* ε-optimal dual-feasible vector y has many non-zero entries, then the number of iterations needs to be large. For example, if every ε-optimal dual-feasible vector y has $\Omega(m)$ non-zero entries, then these methods require $T = \Omega(m)$ iterations before they can reach an ε-optimal dual-feasible vector. Since $T = \mathcal{O}\Big(\frac{R^2r^2}{\varepsilon^2}\ln(n)\Big)$ this would imply that $\frac{Rr}{\varepsilon} = \Omega(\sqrt{m/\ln(n)})$, and hence many classical SDP-solvers would have a better complexity than our quantum SDP-solver. As we show in Section 3, this will actually be the case for families of SDPs that have a lot of symmetry.

1.4.2 Tools that may be of more general interest

Along the way to our improved SDP-solver, we developed some new techniques that may be of independent interest. These are mostly tucked away in appendices, but here we will highlight two.

Implementing smooth functions of a given Hamiltonian. Let $f : \mathbb{R} \to \mathbb{C}$ be a function. In Appendix B we describe a general technique to apply a function $f(H)$ of a sparse Hamilto-

[4] Independently of us, Ben-David, Eldar, Garg, Kothari, Natarajan, and Wright (at MIT), and separately Ambainis observed that in the special case where all b_j are at least 1, the oracle can even be made 1-sparse, and the one entry can be found using one Grover search over m points (in both cases personal communication 2017). The same happens implicitly in our Section 2.3 in this case. However, in general 2 non-zero entries are necessary in y.

nian H to a given state $|\phi\rangle$.[5] Roughly speaking, what applying $f(H)$ to $|\phi\rangle$ means, is that we want a unitary circuit that maps $|0\rangle|\phi\rangle$ to $|0\rangle f(H)|\phi\rangle + |1\rangle|*\rangle$. If need be, we can then combine this with amplitude amplification to boost the $|0\rangle f(H)|\phi\rangle$ part of the state. If the function f can be approximated well by a low-degree Fourier series, then our preparation will be efficient in the sense of using few queries to H and few other gates. The novelty of our approach is that we construct a good Fourier series from the polynomial that approximates f (for example a truncated Taylor series for f). Our Theorem 40 can be easily applied to various smooth functions without using involved integral approximations, unlike previous works building on similar techniques. Our most general result, Corollary 42, only requires that the function f can be nicely approximated locally around each possible eigenvalue of H, improving on Theorem 40.

In this paper we mostly care about the function $f(x) = e^{-x}$, which is what we want for generating a purification of the Gibbs state corresponding to H; and the function $f(x) = \sqrt{x}$, which is what we use for estimating quantities like $\mathrm{Tr}(A\rho)$. However, our techniques apply much more generally than to these two functions. For example, they also simplify the analysis of the improved linear-systems solver of Childs et al. [CKS17], where the relevant function is $f(x) = 1/x$. As in their work, the Linear Combination of Unitaries technique of Childs et al. [CW12, BCC+15, BCK15] is a crucial tool for us.

A generalized minimum-finding algorithm. Dürr and Høyer [DH96] showed how to find the minimal value of a function $f : [N] \to \mathbb{R}$ using $\mathcal{O}(\sqrt{N})$ queries to f, by repeatedly using Grover search to find smaller and smaller elements of the range of f. In Appendix C we describe a more general minimum-finding procedure. Suppose we have a unitary U which prepares a quantum state $U|0\rangle = \sum_{k=1}^{N} |\psi_k\rangle|x_k\rangle$, where the $|\psi_k\rangle$ are unnormalized states. Our procedure can find the minimum value x_{k^*} among the x_k's that have support in the second register, using roughly $\mathcal{O}(1/\||\psi_{k^*}\rangle\|)$ applications of U and U^{-1}. Also, upon finding the minimal value k^* the procedure actually outputs the normalized state proportional to $|\psi_{k^*}\rangle|x_{k^*}\rangle$. This immediately gives the Dürr-Høyer result as a special case, if we take U to produce $U|0\rangle = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} |k\rangle|f(k)\rangle$ using one query to f. Unlike Dürr-Høyer, we need not assume direct query access to the individual values $f(k)$.

More interestingly for us, for a given n-dimensional Hamiltonian H, if we combine our minimum-finder with phase estimation using unitary $U = e^{iH}$ on one half of a maximally entangled state, then we obtain an algorithm for estimating the smallest eigenvalue of H (and preparing its ground state) using roughly $\mathcal{O}(\sqrt{n})$ applications of phase estimation with U. A similar result on approximating the smallest eigenvalue of a Hamiltonian was already shown by Poulin and Wocjan [PW09a], but we improve on their analysis to be able to apply it as a subroutine in our procedure to estimate $\mathrm{Tr}(A_j\rho)$.

1.4.3 Lower bounds

What about lower bounds for quantum SDP-solvers? Brandão and Svore already proved that a quantum SDP-solver has to make $\Omega(\sqrt{n} + \sqrt{m})$ queries to the input matrices, for some SDPs. Their lower bound is for a family of SDPs where $s, R, r, 1/\varepsilon$ are all constant, and is by reduction from a search problem.

In this paper we prove lower bounds that are quantitatively stronger in m and n, but for SDPs with non-constant R and r. The key idea is to consider a Boolean function F on $N = abc$

[5] Here a univariate function $f : \mathbb{R} \to \mathbb{C}$ is applied to a Hermitian matrix H in the standard way, by acting on the eigenvalues of H: if H has diagonalization $H = U^{-1}DU$, with D the diagonal matrix of H's eigenvalues, then $f(H)$ is the matrix $U^{-1}f(D)U$, where the diagonal matrix $f(D)$ is obtained from D by applying f to its diagonal entries.

input bits that is the composition of an a-bit majority function with a b-bit OR function with a c-bit majority function. The known quantum query complexities of majority and OR, combined with composition properties of the adversary lower bound, imply that every quantum algorithm that computes this function requires $\Omega(a\sqrt{b}\,c)$ queries. We define a family of LPs with non-constant R and r such that a constant-error approximation of OPT computes F. Choosing a, b, and c appropriately, this implies a lower bound of

$$\Omega\left(\sqrt{\max\{n,m\}}(\min\{n,m\})^{3/2}\right)$$

queries to the entries of the input matrices for quantum LP-solvers. Since LPs are SDPs with sparsity $s = 1$, we get the same lower bound for quantum SDP-solvers. If m and n are of the same order, this lower bound is $\Omega(mn)$, the same scaling with mn as the classical general instantiation of Arora-Kale (3). In particular, this shows that we cannot have an $O(\sqrt{mn})$ upper bound without simultaneously having polynomial dependence on Rr/ε. The construction of our lower bound implies that for the case $m \approx n$, this polynomial dependence has to be at least $(Rr/\varepsilon)^{1/4}$.

Subsequent work. Following the first version of our paper, improvements in the runtime of quantum SDP-solvers were obtained in [BKL+19, vAG19a], the latter providing a runtime of $\widetilde{\mathcal{O}}\left((\sqrt{m}+\sqrt{n}\frac{Rr}{\epsilon})s\left(\frac{Rr}{\epsilon}\right)^4\right)$. For the special case of LP-solving, where $s = 1$, [vAG19b] further improved the runtime to $\widetilde{\mathcal{O}}\left((\sqrt{m}+\sqrt{n})\left(\frac{Rr}{\epsilon}\right)^3\right)$.

In a different algorithmic direction, Kerenidis and Prakash [KP18] recently obtained a quantum interior-point method for solving SDPs and LPs. It is hard to compare the latter algorithm to the other SDP-solvers for two reasons. First, the output of their algorithm consists only of almost-feasible solutions to the primal and dual (their algorithm has a polynomial dependence on the distance to feasibility). It is therefore not clear what their output means for the optimal value of the SDPs. Secondly, the runtime of their algorithm depends polynomially on the condition number of the matrices that the interior point method encounters, and no explicit bounds for these condition numbers are given.

Our results on implementing smooth functions of a given Hamiltonian have been extended to more general input models (block-encodings) in [GSLW19]. This recent paper builds on some of our techniques, but achieves slightly improved complexities by directly implementing the transformations without using Hamiltonian simulation as a subroutine.

Recently van Apeldoorn et al. [vAGGdW20] and Chakrabarti et al. [CCLW20] developed quantum algorithms for general black-box convex optimization, where one optimizes over a general convex set K, and the access to K is via membership and/or separation oracles. Since we work in a model where we are given access directly to the constraints defining the problem, our results are incomparable to theirs.

Organization. The paper is organized as follows. In Section 2 we start with a description of the Arora-Kale framework for SDP-solvers, and then we describe how to quantize different aspects of it to obtain a quantum SDP-solver with better dependence on R, r, and $1/\varepsilon$ (or rather, on Rr/ε) than Brandão and Svore got. In Section 3 we describe the limitations of primal-dual SDP-solvers using general oracles (not optimized for specific SDPs) that produce sparse dual solutions y: if good solutions are dense, this puts a lower bound on the number of iterations needed. In Section 4 we give our lower bounds. A number of the proofs are relegated to the appendices: how to classically approximate $\mathrm{Tr}(A_j\rho)$ (Appendix A), how to

efficiently implement smooth functions of Hamiltonians on a quantum computer (Appendix B), our generalized method for minimum-finding (Appendix C), upper and lower bounds on how efficiently we can query entries of sums of sparse matrices (Appendix D), and how to trade off the parameters R, r, and $1/\varepsilon$ against each other (Appendix E).

2 An improved quantum SDP-solver

Here we describe our quantum SDP-solver. In Section 2.1 we describe the framework designed by Arora and Kale for solving semidefinite programs. As in the recent work by Brandão and Svore, we use this framework to design an efficient quantum algorithm for solving SDPs. In particular, we show that the key subroutine needed in the Arora-Kale framework can be implemented efficiently on a quantum computer. Our implementation uses different techniques than the quantum algorithm of Brandão and Svore, allowing us to obtain a faster algorithm. The techniques required for this subroutine are developed in Sections 2.2 and 2.3. In Section 2.4 we put everything together to prove the main theorem of this section (the notation is explained below):

Theorem 1. *Instantiating Meta-Algorithm 1 using the trace calculation algorithm from Section 2.2 and the oracle from Section 2.3 (with width-bound $w := r + 1$), and using this to do a binary search for* $\mathrm{OPT} \in [-R, R]$ *(using different guesses α for* OPT*), gives a quantum algorithm for solving SDPs of the form (1), which (with high probability) produces a feasible solution y to the dual program which is optimal up to an additive error ε, and uses*

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

queries to the input matrices and the same order of other gates.

Notation/Assumptions. We use log to denote the logarithm in base 2. We denote the all-zero matrix and vector by 0 and the all-one vector by $\mathbf{1}$. Throughout we assume each element of the input matrices can be represented by a bitstring of size $\mathrm{poly}(\log n, \log m)$. We use s as the sparsity of the input matrices, that is, the maximum number of non-zero entries in a row (or column) of any of the matrices $C, A_1, \ldots, A_m$ is s. Recall that for normalization purposes we assume $\|A_1\|, \ldots, \|A_m\|, \|C\| \leq 1$. We furthermore assume that $A_1 = I$ and $b_1 = R$, that is, the trace of primal-feasible solutions is bounded by R (and hence also the trace of primal-optimal solutions is bounded by R). The analogous quantity for the dual SDP (2), an upper bound on $\sum_{j=1}^m y_j$ for an optimal dual solution y, will be denoted by r. However, we do not add the constraint $\sum_{j=1}^m y_j \leq r$ to the dual. We will assume $r \geq 1$. For r to be well-defined we have to make the explicit assumption that the optimal solution in the dual is attained. In Section 3 it will be necessary to work with the best possible upper bounds: we let R^* be the smallest trace of an optimal solution to SDP (1), and we let r^* be the smallest ℓ_1-norm of an optimal solution to the dual. These quantities are well-defined; indeed, both the primal and dual optimum are attained: the dual optimum is attained by assumption, and due to the assumption $A_1 = I$, the dual SDP is strictly feasible, which means that the optimum in (1) is attained.

Unless specified otherwise, we always consider *additive* error. In particular, an ε-optimal solution to an SDP will be a feasible solution whose objective value is within additive error ε of the optimum.

Input oracles: We assume sparse black-box access to the elements of matrices $C, A_1, \ldots, A_m$ defined in the following way: for input $(j, k, \ell) \in (\{0\} \cup [m]) \times [n] \times [s]$ we can query the location

and value of the ℓth non-zero entry in the kth row of the matrix A_j (where $j = 0$ would indicate the C matrix).

Specifically in the quantum case, as described in [BCK15], we assume access to an oracle O_I which serves the purpose of sparse access. O_I calculates the $\text{index}_{A_j} : [n] \times [s] \to [n]$ function, which for input (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":

$$O_I|j, k, \ell\rangle = |j, k, \text{index}_{A_j}(k, \ell)\rangle. \tag{4}$$

(In the degenerate case where the kth row has fewer than ℓ non-zero entries, $\text{index}_{A_j}(k, \ell)$ is defined to be ℓ together with some special symbol.) We also assume we can apply the inverse of O_I.

We also need another oracle O_M, returning a bitstring representation of $(A_j)_{ki}$ for any $j \in \{0\} \cup [m]$ and $k, i \in [n]$:

$$O_M|j, k, i, z\rangle = |j, k, i, z \oplus (A_j)_{ki}\rangle. \tag{5}$$

The slightly unusual "in place" definition of oracle O_I is not too demanding. In particular, if instead we had an oracle that computed the non-zero entries of a row in order, then we could implement both O_I and its inverse using $\log(s)$ queries (we can compute ℓ from k and $\text{index}_{A_j}(k, \ell)$ using binary search) [BCK15].

Computational model: As our computational model, we assume a slight relaxation of the usual quantum circuit model: a classical control system that can run quantum subroutines. We limit the classical control system so that its number of operations is at most a polylogarithmic factor bigger than the gate complexity of the quantum subroutines, i.e., if the quantum subroutines use C gates, then the classical control system may not use more than $\mathcal{O}(C \operatorname{polylog}(C))$ operations.

When we talk about gate complexity, we count the number of two-qubit quantum gates needed for implementation of the quantum subroutines. Additionally, we assume for simplicity that there exists a unit-cost QRAM gate that allows us to store and retrieve qubits in a memory, by means of a swap of two registers indexed by another register:

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

where the registers $r_1, \ldots, r_K$ are only accessible through this gate. The QRAM gate can be seen as a quantum analogue of pointers in classical computing. The only place where we need QRAM is in Appendix D, for a data structure that allows efficient access to the non-zero entries of a sum of sparse matrices; for the special case of LP-solving it is not needed.

2.1 The Arora-Kale framework for solving SDPs

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 $\text{Tr}(M\rho)$. 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 1 we state the MMW algorithm, the following theorem shows the key property of the output of the algorithm.

Input Parameter $\eta \leq 1$, number of rounds T.

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

Output A sequence of symmetric $n \times n$ matrices $M^{(1)}, \ldots, M^{(T)}$ s.t. $-I \preceq M^{(t)} \preceq I$, 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:

$$\rho^{(t+1)} := \exp\left(-\eta \sum_{\tau=1}^{t} M^{(\tau)}\right) \Big/ \mathrm{Tr}\left(\exp\left(-\eta \sum_{\tau=1}^{t} M^{(\tau)}\right)\right)$$

Algorithm 1: Matrix Multiplicative Weights (MMW) Algorithm

Theorem 2 ([AK16, Theorem 3.1])**.** *For every adversary, the sequence $\rho^{(1)}, \ldots, \rho^{(T)}$ of density matrices constructed using the Matrix Multiplicative Weights Algorithm (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, the MMW algorithm is used to build a vector $y \geq 0$ such that

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

and $b^T y \leq \alpha$. 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 α. 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 $b^T\overline{y} \leq \alpha + \varepsilon$. In order to present the algorithm formally, we require some definitions.

Given a candidate solution $X \succeq 0$ for the primal problem (1) and a parameter $\varepsilon \geq 0$, define

the polytope

$$\mathcal{P}_\varepsilon(X) := \{y \in \mathbb{R}^m : b^T y \leq \alpha, \quad \mathrm{Tr}\left(\Big(\sum_{j=1}^m y_j A_j - C\Big)X\right) \geq -\varepsilon, \quad y \geq 0\}.$$

One can verify the following:

Lemma 3 ([AK16, Lemma 4.2])**.** *If for a given candidate solution $X \succeq 0$ the polytope $\mathcal{P}_0(X)$ is empty, then a scaled version of X is primal-feasible and of objective value at least α.*

The Arora-Kale framework for solving SDPs uses the MMW algorithm where the role of the adversary is taken by an ε-approximate oracle:

Input An $n \times n$ psd matrix X, a number $\alpha \in [-R, R]$, and the description of an SDP as in (1).

Output Either the $\mathsf{Oracle}_\varepsilon$ returns a vector y from the polytope $\mathcal{P}_\varepsilon(X)$ or it outputs "fail". It may only output fail if $\mathcal{P}_0(X) = \emptyset$.

Algorithm 2: Definition of an ε-approximate $\mathsf{Oracle}_\varepsilon$ for maximization SDPs

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

Definition 4 (*Width* of $\mathsf{Oracle}_\varepsilon$)**.** *The* width *of $\mathsf{Oracle}_\varepsilon$ for an SDP is the smallest $w^* \geq 0$ such that for every $X \succeq 0$ and $\alpha \in [-R, R]$, the vector y returned by $\mathsf{Oracle}_\varepsilon$ satisfies $\left\|\sum_{j=1}^m y_j A_j - C\right\| \leq w^*$.*

In practice, the width of an oracle is not always known. However, it suffices to work with an upper bound $w \geq w^*$: as we can see in Meta-Algorithm 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 1:

Theorem 5 ([AK16, Theorem 4.7])**.** *Suppose we are given an SDP of the form (1) with input matrices $A_1 = I, A_2, \ldots, A_m$ and C having operator norm at most 1, and input reals $b_1 = R, b_2, \ldots, b_m$. Assume Meta-Algorithm 1 does not output "fail" in any of the rounds, then the returned vector $\overline{y}$ is feasible for the dual (2) with objective value at most $\alpha + \varepsilon$. If $\mathsf{Oracle}_{\varepsilon/3}$ outputs "fail" in the t-th round then a suitably scaled version of $X^{(t)}$ is primal-feasible with objective value at least α.*

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 $-\eta H^{(t)}$ and the application of the oracle. Note that the only purpose of computing the matrix exponential is to allow the oracle to compute the values $\mathrm{Tr}(A_j X)$ for all j and $\mathrm{Tr}(CX)$, since the polytope depends on X only through those values. 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) oracle with approximations of $\mathrm{Tr}(A_j X)$. Let $a_j := \mathrm{Tr}(A_j \rho) =$

Input The input matrices and reals of SDP (1) and trace bound R. The current guess α of the optimal value of the dual (2). An additive error tolerance $\varepsilon > 0$. An $\frac{\varepsilon}{3}$-approximate oracle $\mathsf{Oracle}_{\varepsilon/3}$ as in Algorithm 2 with width-bound w.

Output Either "Lower" and a vector $\overline{y} \in \mathbb{R}^m_+$ feasible for (2) with $b^T\overline{y} \leq \alpha + \varepsilon$
or "Higher" and a symmetric $n \times n$ matrix X that, when scaled suitably, is primal-feasible with objective value at least α.

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

Meta-Algorithm 1: Primal-Dual Algorithm for solving SDPs

$\mathrm{Tr}(A_jX)/\mathrm{Tr}(X)$ and $c := \mathrm{Tr}(C\rho) = \mathrm{Tr}(CX)/\mathrm{Tr}(X)$. Then, given a list of reals $\tilde{a}_1, \ldots, \tilde{a}_m, \tilde{c}$ and a parameter $\theta \geq 0$, such that $|\tilde{a}_j - a_j| \leq \theta$ for all j, and $|\tilde{c} - c| \leq \theta$, we define the polytope

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

For convenience we will denote $\tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_m)$ and $c' := \tilde{c} - (r+1)\theta$. Notice that $\tilde{\mathcal{P}}$ 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 (2) with $\|y\|_1 \leq r$. Hence, using that $\mathcal{P}_0(X)$ is a *relaxation* of the feasible region of the dual (with bound α on the objective value), we may restrict our oracle to return only such y:

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

The benefit of this restriction is that an oracle 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 3 does not hold for $\mathcal{P}_0(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$.[6]

The following shows that an oracle that always returns a vector $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$ if one exists, is a $4Rr\theta$-approximate oracle as defined in Algorithm 2.

Lemma 6. *Let $\rho = X/\mathrm{Tr}(X)$ where $\mathrm{Tr}(X) \leq R$. Let $\tilde{a}_1, \ldots, \tilde{a}_m$ and $\tilde{c}$ be θ-approximations of $\mathrm{Tr}(A_1\rho), \ldots, \mathrm{Tr}(A_m\rho)$ and $\mathrm{Tr}(C\rho)$ respectively. Then the following holds:*

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

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

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

where in the first inequality we subtracted $\sum_{j=1}^m \mathrm{Tr}(A_j\rho)y_j - \mathrm{Tr}(C\rho) \geq 0$.

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

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

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

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

For the latter inequality we used $\mathrm{Tr}(X) \leq R$. □

We have now seen the Arora-Kale framework for solving SDPs. To obtain a quantum SDP-solver it remains to provide a quantum oracle subroutine. By the above discussion it suffices to set $\theta = \varepsilon/(12Rr)$ and to use an oracle that is based on θ-approximations of $\mathrm{Tr}(A\rho)$ (for $A \in \{A_1, \ldots, A_m, C\}$), since with that choice of θ we have $\mathcal{P}_{4Rr\theta}(X) = \mathcal{P}_{\varepsilon/3}(X)$. In the section below we first give a quantum algorithm for approximating $\mathrm{Tr}(A\rho)$ efficiently (see also Appendix A for a classical algorithm). Then, in Section 2.3, we provide an oracle using those estimates. The oracle will be based on a simple geometric idea and can be implemented both on a quantum computer and on a classical computer (of course, resulting in different runtimes). In Section 2.4 we conclude with an overview of the runtime of our quantum SDP-solver. We want to stress that our solver is meant to work for any SDP. In particular, our oracle does not use the structure of a specific SDP. As we will show in Section 3, any oracle that works for all SDPs necessarily has a large width-bound. To obtain quantum speedups for a *specific* class of SDPs it will be necessary to develop oracles 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 oracles.

[6] Using several transformations of the SDP, from Appendix E and Lemma 2 of [BS17], one can show that there is a way to remove the need for this restriction. Hence, after these modifications, if for a given candidate solution $X \succeq 0$ the oracle outputs that the set $\mathcal{P}_0(X)$ is empty, then a scaled version of X is primal feasible for this new SDP, with objective value at least α. This scaled version of X can be modified to a near-feasible solution to the original SDP (it will be psd, but it might violate the linear constraints a little bit) with nearly the same objective value.

2.2 Approximating the expectation value $\mathrm{Tr}(A\rho)$ using a quantum algorithm

In this section we give an efficient quantum algorithm to approximate quantities of the form $\mathrm{Tr}(A\rho)$. We are going to work with Hermitian matrices $A, H \in \mathbb{C}^{n\times n}$, such that ρ is the Gibbs state $e^{-H}/\mathrm{Tr}\left(e^{-H}\right)$. Note the analogy with quantum physics: in physics terminology $\mathrm{Tr}(A\rho)$ is simply called the "expectation value" of A for a quantum system in a thermal state corresponding to H.

The general approach is to separately estimate $\mathrm{Tr}\left(Ae^{-H}\right)$ and $\mathrm{Tr}\left(e^{-H}\right)$, and then to use the ratio of these estimates as an approximation of $\mathrm{Tr}(A\rho) = \mathrm{Tr}\left(Ae^{-H}\right)/\mathrm{Tr}\left(e^{-H}\right)$. Both estimations are done using state preparation to prepare a pure state with a flag, such that the probability that the flag is 0 is proportional to the quantity we want to estimate, and then to use amplitude estimation to estimate that probability. Below in Section 2.2.1 we first describe the general approach. In Section 2.2.2 we then instantiate this for the special case where all matrices are diagonal, which is the relevant case for LP-solving. In Section 2.2.3 we handle the general case of arbitrary matrices (needed for SDP-solving); the state-preparation part will be substantially more involved there, because in the general case we need not know the diagonalizing bases for A and H, and A and H may not be simultaneously diagonalizable.

2.2.1 General approach

To start, consider the following lemma about the multiplicative approximation error of a ratio of two real numbers that are given by multiplicative approximations:

Lemma 7. *Let $0 \le \theta \le 1$ and let $\alpha, \tilde{\alpha}, Z, \tilde{Z}$ be positive real numbers such that $|\alpha - \tilde{\alpha}| \le \alpha\theta/3$ and $|Z - \tilde{Z}| \le Z\theta/3$. Then*

$$\left|\frac{\alpha}{Z} - \frac{\tilde{\alpha}}{\tilde{Z}}\right| \le \theta\frac{\alpha}{Z}$$

Proof. The inequality can be proven as follows

$$\left|\frac{\alpha}{Z} - \frac{\tilde{\alpha}}{\tilde{Z}}\right| = \left|\frac{\alpha\tilde{Z} - \tilde{\alpha}Z}{Z\tilde{Z}}\right| = \left|\frac{\alpha\tilde{Z} - \alpha Z + \alpha Z - \tilde{\alpha}Z}{Z\tilde{Z}}\right| \le \left|\frac{\alpha\tilde{Z} - \alpha Z}{Z\tilde{Z}}\right| + \left|\frac{\alpha Z - \tilde{\alpha}Z}{Z\tilde{Z}}\right| \le \frac{\alpha\theta}{3\tilde{Z}} + \frac{\alpha\theta}{3\tilde{Z}} \le \theta\frac{\alpha}{Z}$$

where the last step used $\tilde{Z} \ge \frac{2}{3}Z$. □

Corollary 8. *Let A be such that $\|A\| \le 1$. A multiplicative $\frac{\theta}{9}$-approximation of both $\mathrm{Tr}\left(\frac{I}{4}e^{-H}\right)$ and $\mathrm{Tr}\left(\frac{I+A/2}{4}e^{-H}\right)$ suffices to get an additive θ-approximation of $\frac{\mathrm{Tr}\left(Ae^{-H}\right)}{\mathrm{Tr}(e^{-H})}$.*

Proof. According to Lemma 7 by dividing the two multiplicative approximations we get

$$\frac{\theta}{3}\frac{\mathrm{Tr}\left(\frac{I+A/2}{4}e^{-H}\right)}{\mathrm{Tr}\left(\frac{I}{4}e^{-H}\right)} = \frac{\theta}{3}\left(1 + \frac{\mathrm{Tr}\left(\frac{A}{2}e^{-H}\right)}{\mathrm{Tr}(e^{-H})}\right) \le \frac{\theta}{3}\left(1 + \frac{\|A\|}{2}\right) \le \theta/2,$$

i.e., an additive $\theta/2$-approximation of

$$1 + \frac{\mathrm{Tr}\left(\frac{A}{2}e^{-H}\right)}{\mathrm{Tr}(e^{-H})},$$

which yields an additive θ-approximation to $\mathrm{Tr}\left(Ae^{-H}\right)/\mathrm{Tr}\left(e^{-H}\right)$. □

It thus suffices to separately approximate both quantities from the corollary. Notice that both are of the form $\mathrm{Tr}\left(\frac{I+A/2}{4}e^{-H}\right)$, the first with the actual A, the second with $A = 0$. Furthermore, a multiplicative $\theta/9$-approximation to both can be achieved by approximating both up to an additive error $\theta\mathrm{Tr}\left(e^{-H}\right)/72$, since $\mathrm{Tr}\left(\frac{I}{8}e^{-H}\right) \leq \mathrm{Tr}\left(\frac{I+A/2}{4}e^{-H}\right)$.

For now, let us assume we can construct a unitary $U_{A,H}$ such that if we apply it to the state $|0\dots0\rangle$ then we get a probability $\frac{\mathrm{Tr}\left((I+A/2)e^{-H}\right)}{4n}$ of outcome 0 when measuring the first qubit. That is:

$$\|(\langle 0|\otimes I)U_{A,H}|0\dots0\rangle\|^2 = \frac{\mathrm{Tr}\left((I+A/2)e^{-H}\right)}{4n}.$$

(To clarify the notation: if $|\psi\rangle$ is a 2-register state, then $(\langle 0|\otimes I)|\psi\rangle$ is the (unnormalized) state in the 2nd register that results from projecting on $|0\rangle$ in the 1st register.)

In practice we will not be able to construct such a $U_{A,H}$ exactly, instead we will construct a $\tilde{U}_{A,H}$ that yields a sufficiently close approximation of the correct probability. When we have access to such a unitary, the following lemma allows us to use amplitude estimation to estimate the probability and hence $\mathrm{Tr}\left(\frac{I+A/2}{4}e^{-H}\right)$ up to the desired error.

Lemma 9. *Suppose we have a unitary U acting on q qubits such that $U|0\dots0\rangle = |0\rangle|\psi\rangle + |\Phi\rangle$ with $(\langle 0|\otimes I)|\Phi\rangle = 0$ and $\|\psi\|^2 = p \geq p_{\min}$ for some known bound $p_{\min}$. Let $\mu \in (0,1]$ be the allowed multiplicative error in our estimation of p. Then with $\mathcal{O}\left(\frac{1}{\mu\sqrt{p_{\min}}}\right)$ uses of U and U^{-1} and using $\mathcal{O}\left(\frac{q}{\mu\sqrt{p_{\min}}}\right)$ gates on the q qubits and some ancilla qubits, we obtain a $\tilde{p}$ such that $|p-\tilde{p}| \leq \mu p$ with probability at least $4/5$.*

Proof. We use the amplitude-estimation algorithm of [BHMT02, Theorem 12] with M applications of U and U^{-1}. This provides an estimate $\tilde{p}$ of p, that with probability at least $8/\pi^2 > 4/5$ satisfies

$$|p-\tilde{p}| \leq 2\pi\frac{\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2} \leq \frac{\pi}{M}\left(2\sqrt{p} + \frac{\pi}{M}\right).$$

Choosing M the smallest power of 2 such that $M \geq 3\pi/(\mu\sqrt{p_{\min}})$, with probability at least $4/5$ we get

$$|p-\tilde{p}| \leq \mu\frac{\sqrt{p_{\min}}}{3}\left(2\sqrt{p} + \mu\frac{\sqrt{p_{\min}}}{3}\right) \leq \mu\frac{\sqrt{p}}{3}(3\sqrt{p}) \leq \mu p.$$

The q factor in the gate complexity comes from the implementation of the amplitude amplification steps needed in amplitude estimation. The gate complexity of the whole amplitude-estimation procedure is dominated by this contribution, proving the final gate complexity. □

Corollary 10. *Suppose we are given the positive numbers $z \leq \mathrm{Tr}\left(e^{-H}\right)$, $\theta \in (0,1]$, and unitary circuits $\tilde{U}_{A',H}$ for $A' = 0$ and $A' = A$ with $\|A\| \leq 1$, each acting on at most q qubits such that*

$$\left|\left\|(\langle 0|\otimes I)\tilde{U}_{A',H}|0\dots0\rangle\right\|^2 - \frac{\mathrm{Tr}\left((I+A'/2)e^{-H}\right)}{4n}\right| \leq \frac{\theta z}{144n}.$$

Applying the procedure of Lemma 9 to $\tilde{U}_{A',H}$ (both for $A' = 0$ and for $A' = A$) with $p_{\min} = \frac{z}{9n}$ and $\mu = \theta/19$, and combining the results using Corollary 8 yields an additive θ-approximation of $\mathrm{Tr}(A\rho)$ with probability at least $4/5$. The procedure uses

$$\mathcal{O}\left(\frac{1}{\theta}\sqrt{\frac{n}{z}}\right)$$

applications of $\tilde{U}_{A,H}$, $\tilde{U}_{0,H}$ and their inverses, and $\mathcal{O}\left(\frac{q}{\theta}\sqrt{\frac{n}{z}}\right)$ additional gates.

Proof. First note that since $I + A'/2 \succeq I/2$, we have

$$t := \frac{\mathrm{Tr}\Big((I + A'/2)e^{-H}\Big)}{4n} \geq \frac{\mathrm{Tr}\Big(e^{-H}\Big)}{8n},$$

and thus

$$\left|\left\|(\langle 0| \otimes I)\tilde{U}_{A',H}|0\dots 0\rangle\right\|^2 - t\right| \leq \frac{\theta z}{144n} \leq \frac{\theta}{18} \cdot \frac{\mathrm{Tr}\Big(e^{-H}\Big)}{8n} \leq \frac{\theta}{18}t \leq \frac{t}{18}. \tag{6}$$

Therefore

$$\left\|(\langle 0| \otimes I)\tilde{U}_{A',H}|0\dots 0\rangle\right\|^2 \geq \left(1 - \frac{1}{18}\right)t \geq \left(1 - \frac{1}{18}\right)\frac{\mathrm{Tr}\Big(e^{-H}\Big)}{8n} > \frac{\mathrm{Tr}\Big(e^{-H}\Big)}{9n} \geq \frac{z}{9n} =: p_{\min}.$$

Also by (6) we have

$$\left\|(\langle 0| \otimes I)\tilde{U}_{A',H}|0\dots 0\rangle\right\|^2 \leq \left(1 + \frac{\theta}{18}\right)t \leq \frac{19}{18}t.$$

Therefore using Lemma 9 and setting $\mu = \theta/19$, with probability at least $4/5$ we get a $\tilde{p}$ satisfying

$$\left|\tilde{p} - \left\|(\langle 0| \otimes I)\tilde{U}_{A',H}|0\dots 0\rangle\right\|^2\right| \leq \frac{\theta}{19} \cdot \left\|(\langle 0| \otimes I)\tilde{U}_{A',H}|0\dots 0\rangle\right\|^2 \leq \frac{\theta}{18}t. \tag{7}$$

By combining (6)-(7) and using the triangle inequality we get

$$|t - \tilde{p}| \leq \frac{\theta}{9}t,$$

so that Corollary 8 can indeed be applied. The complexity statement follows from Lemma 9 and our choices of $p_{\min}$ and μ. □

Notice the $1/\sqrt{z} \geq 1/\sqrt{\mathrm{Tr}(e^{-H})}$ factor in the complexity statement of the last corollary. To make sure this factor is not too large, we would like to ensure $\mathrm{Tr}\Big(e^{-H}\Big) = \Omega(1)$. This can be achieved by substituting $H_+ = H - \lambda_{\min} I$, where $\lambda_{\min}$ is the smallest eigenvalue of H. It is easy to verify that this will not change the value $\mathrm{Tr}\Big(Ae^{-H}/\mathrm{Tr}\Big(e^{-H}\Big)\Big)$.

It remains to show how to compute $\lambda_{\min}$ and how to apply $\tilde{U}_{A,H}$. Both of these steps are considerably easier in the case where all matrices are diagonal, so we will consider this case first.

2.2.2 The special case of diagonal matrices – for LP-solving

In this section we consider diagonal matrices, assuming oracle access to H of the following form:

$$O_H|i\rangle|z\rangle = |i\rangle|z \oplus H_{ii}\rangle$$

and similarly for A. Notice that this kind of oracle can easily be constructed from the general sparse matrix oracle (5) that we assume access to.

Lemma 11. *Let $A, H \in \mathbb{R}^{n\times n}$ be diagonal matrices such that $\|A\| \leq 1$ and $H \succeq 0$, and let $\mu > 0$ be an error parameter. Then there exists a unitary $\tilde{U}_{A,H}$ such that*

$$\left|\left\|(\langle 0| \otimes I)\tilde{U}_{A,H}|0\dots 0\rangle\right\|^2 - \mathrm{Tr}\left(\frac{I + A/2}{4n}e^{-H}\right)\right| \leq \mu,$$

which uses 1 quantum query to A and H and $\mathcal{O}(\log^{\mathcal{O}(1)}(1/\mu) + \log(n))$ other gates.

Proof. First we prepare the state $\sum_{i=1}^{n} |i\rangle/\sqrt{n}$ with $\mathcal{O}(\log(n))$ one- and two-qubit gates. If n is a power of 2 we do this by applying $\log_2(n)$ Hadamard gates on $|0\rangle^{\otimes \log_2(n)}$; in the general case it is still possible to prepare the state $\sum_{i=1}^{n} |i\rangle/\sqrt{n}$ with $\mathcal{O}(\log(n))$ two-qubit gates, for example by preparing the state $\sum_{i=1}^{k} |i\rangle/\sqrt{k}$ for $k = 2^{\lceil \log_2(n) \rceil}$ and then using (exact) amplitude amplification in order to remove the $i > n$ from the superposition.

Then we query the diagonal values of H and A to get the state $\sum_{i=1}^{n} |i\rangle|H_{ii}\rangle|A_{ii}\rangle/\sqrt{n}$. Using these binary values we apply a finite-precision arithmetic circuit to prepare

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle|H_{ii}\rangle|A_{ii}\rangle|\beta_i\rangle, \text{ where } \beta_i := \arcsin\left(\sqrt{\frac{1 + A_{ii}/2}{4} e^{-H_{ii}} + \delta_i}\right)/\pi, \text{ and } |\delta_i| \leq \mu.$$

The error δ_i is because we only write down a finite number of bits $b_1.b_2b_3 \ldots b_{\log(8/\mu)}$. Due to our choice of A and H, we know that β_i lies in $[0, 1]$. We proceed by first adding an ancilla qubit initialized to $|1\rangle$ in front of the state, then we apply $\log(8/\mu)$ controlled rotations to this qubit: for each $b_j = 1$ we apply a rotation by angle $\pi 2^{-j}$. In other words, if $b_1 = 1$, then we rotate $|1\rangle$ fully to $|0\rangle$. If $b_2 = 1$, then we rotate halfway, and we proceed further by halving the angle for each subsequent bit. We will end up with the state:

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \left(\sqrt{\frac{1 + A_{ii}/2}{4} e^{-H_{ii}} + \delta_i} \, |0\rangle + \sqrt{1 - \frac{1 + A_{ii}/2}{4} e^{-H_{ii}} - \delta_i} \, |1\rangle \right) |i\rangle|A_{ii}\rangle|H_{ii}\rangle|\beta_i\rangle.$$

It is now easy to see that the squared norm of the $|0\rangle$-part of this state is as required:

$$\left\| \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \sqrt{\frac{1 + A_{ii}/2}{4e^{H_{ii}}} + \delta_i} \, |i\rangle \right\|^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1 + A_{ii}/2}{4} e^{-H_{ii}} + \delta_i \right) = \frac{\mathrm{Tr}\left((I + A/2)e^{-H}\right)}{4n} + \sum_{i=1}^{n} \frac{\delta_i}{n},$$

which is an additive μ-approximation since $\left| \sum_{i=1}^{n} \frac{\delta_i}{n} \right| \leq \mu$. □

Corollary 12. *For $A, H \in \mathbb{R}^{n \times n}$ diagonal matrices with $\|A\| \leq 1$, an additive θ-approximation of*

$$\mathrm{Tr}(A\rho) = \frac{\mathrm{Tr}\left(Ae^{-H}\right)}{\mathrm{Tr}(e^{-H})}$$

can be computed using $\mathcal{O}\left(\frac{\sqrt{n}}{\theta}\right)$ queries to A and H, and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\theta}\right)$ other gates.

Proof. Since H is a diagonal matrix, its eigenvalues are exactly its diagonal entries. Using the quantum minimum-finding algorithm of Dürr and Høyer [DH96] one can find (with high success probability) the minimum $\lambda_{\min}$ of the diagonal entries using $\mathcal{O}(\sqrt{n})$ queries to the matrix elements. Applying Lemma 11 and Corollary 10 to $H_+ = H - \lambda_{\min} I$, with $z = 1$, gives the stated bound. □

2.2.3 General case – for SDP-solving

In this section we will extend the ideas from the last section to non-diagonal matrices. There are a few complications that arise in this more general case. These mostly follow from the fact that we now do not know the eigenvectors of H and A, which were the basis states before, and that these eigenvectors might not be the same for both matrices. For example, to find the minimal eigenvalue of H, we can no longer simply minimize over its diagonal entries. To solve this, in Appendix C we develop new techniques that generalize minimum-finding.

Furthermore, the unitary $\tilde{U}_{A,H}$ in the LP case could be seen as applying the operator

$$\sqrt{\frac{I + A/2}{4} e^{-H}}$$

to a superposition of its eigenvectors. This is also more complicated in the general setting, due to the fact that the eigenvectors are no longer the basis states. In Appendix B we develop general techniques to apply smooth functions of Hamiltonians to a state. Among other things, this will be used to create an efficient purified Gibbs sampler.

Our Gibbs sampler uses similar methods to the work of Chowdhury and Somma [CS17] for achieving logarithmic dependence on the precision. However, the result of [CS17] cannot be applied to our setting, because it implicitly assumes access to an oracle for $\sqrt{H}$ instead of H. Although their paper describes a way to construct such an oracle, it comes with a large overhead: they construct an oracle for $\sqrt{H'} = \sqrt{H + \nu I}$, where $\nu \in \mathbb{R}_+$ is some possibly large positive number. This shifting can have a huge effect on $Z' = \mathrm{Tr}\left(e^{-H'}\right) = e^{-\nu}\mathrm{Tr}\left(e^{-H}\right)$, which can be prohibitive due to the $\sqrt{1/Z'}$ factor in the runtime, which blows up exponentially in ν.

In the following lemma we show how to implement $\tilde{U}_{A,H}$ using the techniques we developed in Appendix B.

Lemma 13. *Let $A, H \in \mathbb{C}^{n\times n}$ be Hermitian matrices such that $\|A\| \leq 1$ and $I \preceq H \preceq KI$ for a known $K \in \mathbb{R}_+$. Assume A is s-sparse and H is d-sparse with $s \leq d$. Let $\mu > 0$ be an error parameter. Then there exists a unitary $\tilde{U}_{A,H}$ such that*

$$\left| \left\| (\langle 0| \otimes I)\tilde{U}_{A,H}|0 \dots 0\rangle \right\|^2 - \mathrm{Tr}\left(\frac{I + A/2}{4n} e^{-H}\right) \right| \leq \mu$$

that uses $\widetilde{\mathcal{O}}(Kd)$ queries to A and H, and the same order of other gates.

Proof. The basic idea is that we first prepare a maximally entangled state $\sum_{i=1}^n |i\rangle|i\rangle/\sqrt{n}$, and then apply the (norm-decreasing) maps $e^{-H/2}$ and $\sqrt{\frac{I+A/2}{4}}$ to the first register. Note that we can assume without loss of generality that $\mu \leq 1$, otherwise the statement is trivial.

Let $\tilde{W}_0 = (\langle 0| \otimes I)\tilde{W}(|0\rangle \otimes I)$ be a $\mu/5$-approximation of the map $e^{-H/2}$ (in operator norm) implemented by using Theorem 43, and let $\tilde{V}_0 = (\langle 0| \otimes I)\tilde{V}(|0\rangle \otimes I)$ be a $\mu/5$-approximation of the map $\sqrt{\frac{I+A/2}{4}}$ implemented by using Theorem 40. We define $\tilde{U}_{A,H} := \tilde{V}\tilde{W}$, noting that there is a hidden $\otimes I$ factor in both $\tilde{V}$ and $\tilde{W}$ corresponding to their respective ancilla qubit. As in the linear programming case, we are interested in the probability p of measuring outcome 00 in the first register (i.e., the two "flag" qubits) after applying $\tilde{U}_{A,H}$. We will analyze this in terms of these operators below.

$$\begin{aligned}
p &:= \left\| (\langle 00| \otimes I)\tilde{U}_{A,H}(|00\rangle \otimes I) \sum_{i=1}^n \frac{|i\rangle|i\rangle}{\sqrt{n}} \right\|^2 \\
&= \left\| \tilde{V}_0\tilde{W}_0 \sum_{i=1}^n \frac{|i\rangle|i\rangle}{\sqrt{n}} \right\|^2 \\
&= \frac{1}{n}\sum_{i=1}^n \langle i|\tilde{W}_0^\dagger\tilde{V}_0^\dagger\tilde{V}_0\tilde{W}_0|i\rangle \\
&= \frac{1}{n}\mathrm{Tr}\left(\tilde{W}_0^\dagger\tilde{V}_0^\dagger\tilde{V}_0\tilde{W}_0\right) \\
&= \frac{1}{n}\mathrm{Tr}\left(\tilde{V}_0^\dagger\tilde{V}_0\tilde{W}_0\tilde{W}_0^\dagger\right) \qquad (8)
\end{aligned}$$

Now we show that the above quantity is a good approximation of

$$\frac{1}{n}\mathrm{Tr}\Big(\frac{I+A/2}{4}e^{-H}\Big). \tag{9}$$

For this we show that $\tilde{V}_0^\dagger \tilde{V}_0 \approx (I+A/2)/4$ and $\tilde{W}_0\tilde{W}_0^\dagger \approx e^{-H}$. To see this, first note that for all matrices $B, \tilde{B}$ with $\|B\| \leq 1$, we have

$$\begin{aligned}
\Big\|B^\dagger B - \tilde{B}^\dagger\tilde{B}\Big\| &= \Big\|(B^\dagger - \tilde{B}^\dagger)B + B^\dagger(B-\tilde{B}) - (B^\dagger-\tilde{B}^\dagger)(B-\tilde{B})\Big\| \\
&\leq \Big\|(B^\dagger - \tilde{B}^\dagger)B\Big\| + \Big\|B^\dagger(B-\tilde{B})\Big\| + \Big\|(B^\dagger-\tilde{B}^\dagger)(B-\tilde{B})\Big\| \\
&\leq \Big\|B^\dagger - \tilde{B}^\dagger\Big\|\|B\| + \Big\|B^\dagger\Big\|\Big\|B-\tilde{B}\Big\| + \Big\|B^\dagger-\tilde{B}^\dagger\Big\|\Big\|B-\tilde{B}\Big\| \\
&\leq 2\Big\|B-\tilde{B}\Big\| + \Big\|B-\tilde{B}\Big\|^2.
\end{aligned}$$

Since $\mu \leq 1$, and hence $2\mu/5 + (\mu/5)^2 \leq \mu/2$, this implies (with $B = e^{-H/2}$ and $\tilde{B} = \tilde{W}_0^\dagger$) that $\Big\|e^{-H} - \tilde{W}_0\tilde{W}_0^\dagger\Big\| \leq \mu/2$, and also (with $B = \sqrt{(I+A/2)/4}$ and $\tilde{B} = \tilde{V}_0$) $\Big\|(I+A/2)/4 - \tilde{V}_0^\dagger\tilde{V}_0\Big\| \leq \mu/2$. Let $\|\cdot\|_1$ denote the trace norm (a.k.a. Schatten 1-norm). Note that for all $C, D, \tilde{C}, \tilde{D}$:

$$\begin{aligned}
\Big|\mathrm{Tr}(CD) - \mathrm{Tr}\Big(\tilde{C}\tilde{D}\Big)\Big| &\leq \Big\|CD - \tilde{C}\tilde{D}\Big\|_1 \\
&= \Big\|(C-\tilde{C})D + C(D-\tilde{D}) - (C-\tilde{C})(D-\tilde{D})\Big\|_1 \\
&\leq \Big\|(C-\tilde{C})D\Big\|_1 + \Big\|C(D-\tilde{D})\Big\|_1 + \Big\|(C-\tilde{C})(D-\tilde{D})\Big\|_1 \\
&\leq \Big\|C-\tilde{C}\Big\|\|D\|_1 + \Big\|D-\tilde{D}\Big\|\Big(\|C\|_1 + \Big\|C-\tilde{C}\Big\|_1\Big).
\end{aligned}$$

Which, in our case (setting $C = (I+A/2)/4$, $D = e^{-H}$, $\tilde{C} = \tilde{V}_0^\dagger\tilde{V}_0$, and $\tilde{D} = \tilde{W}_0\tilde{W}_0^\dagger$) implies that

$$\Big|\mathrm{Tr}\Big((I+A/2)e^{-H}/4\Big) - \mathrm{Tr}\Big(\tilde{V}_0^\dagger\tilde{V}_0\tilde{W}_0\tilde{W}_0^\dagger\Big)\Big| \leq (\mu/2)\mathrm{Tr}\Big(e^{-H}\Big) + (\mu/2)(1/2+\mu/2)n.$$

Dividing both sides by n and using equation (8) then implies

$$\begin{aligned}
\Big|\mathrm{Tr}\Big((I+A/2)e^{-H}\Big)/(4n) - p\Big| &\leq \frac{\mu}{2}\frac{\mathrm{Tr}\Big(e^{-H}\Big)}{n} + \frac{\mu}{2}\Big(\frac{1}{2}+\frac{\mu}{2}\Big) \\
&\leq \frac{\mu}{2} + \frac{\mu}{2} \\
&= \mu.
\end{aligned}$$

This proves the correctness of $\tilde{U}_{A,H}$. It remains to show that the complexity statement holds. To show this we only need to specify how to implement the map $\sqrt{\frac{I+A/2}{4}}$ using Theorem 40 (see Appendix B), since the map $e^{H/2}$ is already dealt with in Theorem 43. To use Theorem 40, we choose $x_0 := 0$, $K := 1$ and $r := 1$, since $\|A\| \leq 1$. Observe that $\sqrt{\frac{1+x/2}{4}} = \frac{1}{2}\sum_{k=0}^{\infty}\binom{1/2}{k}\big(\frac{x}{2}\big)^k$ whenever $|x| \leq 1$. Also let $\delta = 1/2$, so $r+\delta = \frac{3}{2}$ and $\frac{1}{2}\sum_{k=0}^{\infty}\Big|\binom{1/2}{k}\Big|\Big(\frac{3}{4}\Big)^k \leq 1 =: B$. Recall that $\tilde{V}$ denotes the unitary that Theorem 40 constructs. Since we choose the precision parameter to be $\mu/5 = \Theta(\mu)$, Theorem 40 shows $\tilde{V}$ can be implemented using $\mathcal{O}\Big(d\log^2(1/\mu)\Big)$ queries and $\mathcal{O}\Big(d\log^2(1/\mu)\Big[\log(n) + \log^{2.5}(1/\mu)\Big]\Big)$ gates. This cost is negligible compared to the cost of our implementation of $e^{-H/2}$ with $\mu/5$ precision: Theorem 43 uses $\mathcal{O}\Big(Kd\log^2(K/\mu)\Big)$ queries and $\mathcal{O}\Big(Kd\log^2(Kd/\mu)\Big[\log(n) + \log^{2.5}(Kd/\mu)\Big]\Big)$ gates to implement $\tilde{W}$. □

Corollary 14. *Let $A, H \in \mathbb{C}^{n\times n}$ be Hermitian matrices such that $\|A\| \leq 1$ and $\|H\| \leq K$ for a known bound $K \in \mathbb{R}_+$. Assume A is s-sparse and H is d-sparse with $s \leq d$. An additive θ-approximation of*

$$\mathrm{Tr}(A\rho) = \frac{Ae^{-H}}{\mathrm{Tr}(e^{-H})}$$

can be computed using $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}dK}{\theta}\right)$ queries to A and H, while using the same order of other gates.

Proof. Start by computing an estimate $\tilde{\lambda}_{\min}$ of $\lambda_{\min}(H)$, the minimum eigenvalue of H, up to additive error $\varepsilon = 1/2$ using Lemma 50. We define $H_+ := H - (\tilde{\lambda}_{\min} - 3/2)I$, so that $I \preceq H_+$ but $2I \not\preceq H_+$.

Applying Lemma 13 and Corollary 10 to H_+ with $z = e^{-2}$ gives the stated bound. □

2.3 An efficient 2-sparse oracle

To motivate the problem below, recall from the end of Section 2.1 that $\tilde{a}_j$ is an additive θ-approximation to $\mathrm{Tr}(A_j\rho)$, $\tilde{c}$ is an additive θ-approximation to $\mathrm{Tr}(C\rho)$ and $c' = \tilde{c} - r\theta - \theta$. We first describe a simplified version of our quantum 2-sparse oracle (Lemma 16) that assumes access to a unitary acting 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 then discuss how to modify the analysis when we are given an oracle that acts as $|j\rangle|0\rangle|0\rangle \mapsto |j\rangle \sum_i \beta_j^i|\tilde{a}_j^i\rangle|\psi_j^i\rangle$ where each $|\tilde{a}_j^i\rangle$ is an approximation of a_j and the amplitudes β_j^i are such that measuring the second register with high probability returns an $\tilde{a}_j^i$ which is θ-close to a_j. We do so since the output of the trace-estimation procedure of the previous section is of this more complicated form.

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

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

Our first observation is that the polytope $\tilde{\mathcal{P}}(\tilde{a}, c')$ 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. The latter follows from general properties of linear programs: any feasible LP with at most k constraints has a k-sparse optimal solution (see, e.g., [Sch86, Ch. 7]). Note that non-emptiness of $\tilde{\mathcal{P}}(\tilde{a}, c')$ is equivalent to the optimal value of

$$\min \quad 1^T y \quad \text{s.t.} \quad b^T y \leq \alpha,\ \tilde{a}^T y \geq c',\ y \geq 0 \tag{10}$$

being at most r (the latter being an LP with only 2 non-trivial constraints). We will give an alternative, constructive proof that we can obtain a 2-sparse solution in Lemma 15. This will also illustrate the intuition behind the definition of our 2-sparse oracle.

Before doing so, let us give a first naive approach to find a $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$ which will not be sufficiently efficient for our purposes. Using the formulation in Equation (10), we could attempt to find a $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$ by solving $\mathcal{O}(m^2)$ linear programs with only 2 variables and 2 constraints each (these LPs are obtained from (10) by setting all but 2 variables equal to zero) and searching for an optimal solution whose value is at most r. Here each linear program is determined by the values $\tilde{a}_j$, b_j and c', and thus we can decide if $\tilde{\mathcal{P}}(\tilde{a}, c')$ is non-empty using $\mathcal{O}(m^2)$ classical time (given these values).

We use a more sophisticated approach to show that $\mathcal{O}(m)$ classical operations (and queries) suffice. Our approach is amenable to a quantum speedup: it also implies that only $\widetilde{\mathcal{O}}(\sqrt{m})$ quantum queries suffice. In particular, we now show how to reduce the problem of finding a $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$ 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 $\alpha \geq 0$ and $c' \leq 0$, then $y = 0$ is a solution and our oracle can return it. From now on we will assume that $\alpha < 0$ 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

$$\begin{aligned} b^T q &\leq \alpha/N \\ \tilde{a}^T q &\geq c'/N \\ \|q\|_1 &= 1 \\ q &\geq 0 \\ 0 < N &\leq r. \end{aligned} \tag{11}$$

We can now view $q \in \mathbb{R}^m_+$ 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 $g_N = (\alpha/N, c'/N)$ for some $0 < N \leq r$. Let $\mathcal{G}_N$ denote the upper-left quadrant of the plane starting at g_N.

Lemma 15. *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 $\|y\|_1 = \|y'\|_1$.*

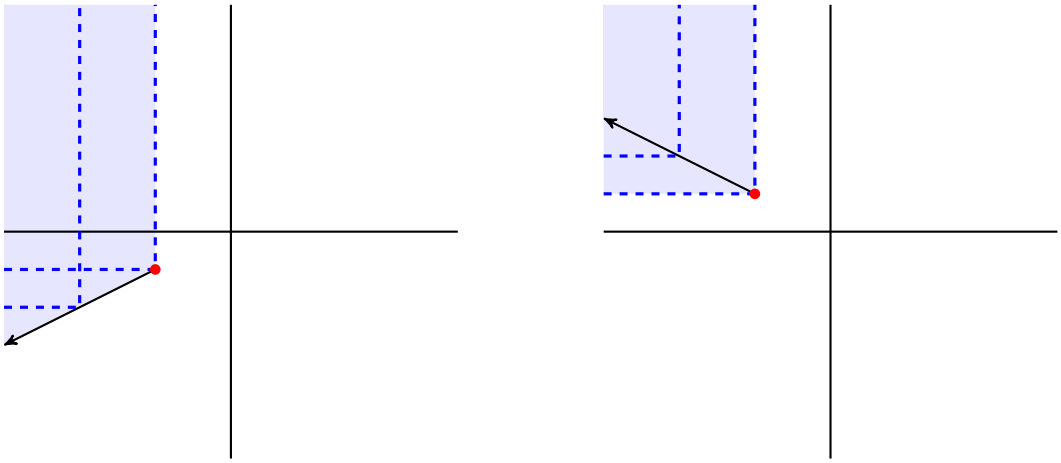
Proof. Let $y \in \tilde{\mathcal{P}}(\tilde{a}, c')$, and $N = \|y\|_1$. Consider $p_i = (b_i, \tilde{a}_i)$ and $g = (\alpha/N, c'/N)$ as before, and write $q = y/N$ such that $\sum_{j=1}^m q_j = 1$, $q \geq 0$. The vector q certifies that a convex combination of the points p_i lies in $\mathcal{G}_N$. 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 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 6 and Lemma 15:

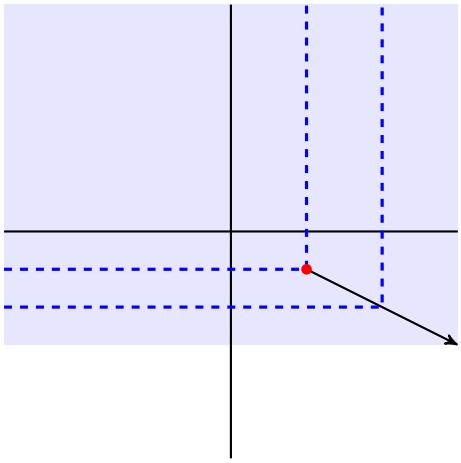
- If $\mathcal{P}_0(X) \cap \{y \in \mathbb{R}^m \colon \sum_j y_j \leq r\}$ is non-empty, then some convex combination of two of the p_j's lies in $\mathcal{G}$.
- If $\mathcal{P}_{4Rr\theta}(X) \cap \{y \in \mathbb{R}^m \colon \sum_j y_j \leq r\}$ is empty, then no convex combination of the p_j's lies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

We first prove a simplified version of the main result of this section. The analysis below 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 later prove the result for more general oracles, which may give superpositions of approximations instead of just the one value $\tilde{a}_j$.

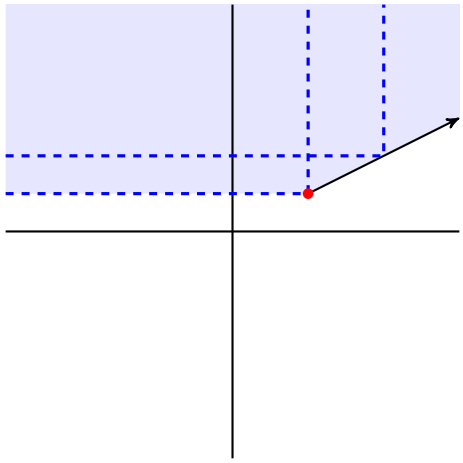


(a) $\alpha < 0, c' < 0$

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



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



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

Figure 1: The region $\mathcal{G}$ in light blue. The borders of two quadrants $\mathcal{G}_N$ have been drawn by thick dashed blue lines. The red dot at the beginning of the arrow is the point $(\alpha/r, c'/r)$.

Lemma 16 (Simple version)**.** *There is an algorithm that returns a 2-sparse vector q (with $q \geq 0$ and $\|q\|_1 = 1$) 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 $\mathcal{O}(m)$ other operations; and a quantum algorithm that (in order to solve the problem with high probability) uses $\mathcal{O}(\sqrt{m})$ calls to an (exact quantum) subroutine that gives the entries of $\tilde{a}$, and $\widetilde{\mathcal{O}}(\sqrt{m})$ other gates.*

Proof. The algorithm can be summarized as follows:

1. Check if $\alpha \geq 0$ 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'.

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 1 for illustration). The shape of $\mathcal{G}$ depends on the sign of α and c'.

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

(b) If $\alpha < 0$ 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 $(\alpha/r, c'/r)$, but now one edge goes up vertically and one goes to the left horizontally.

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

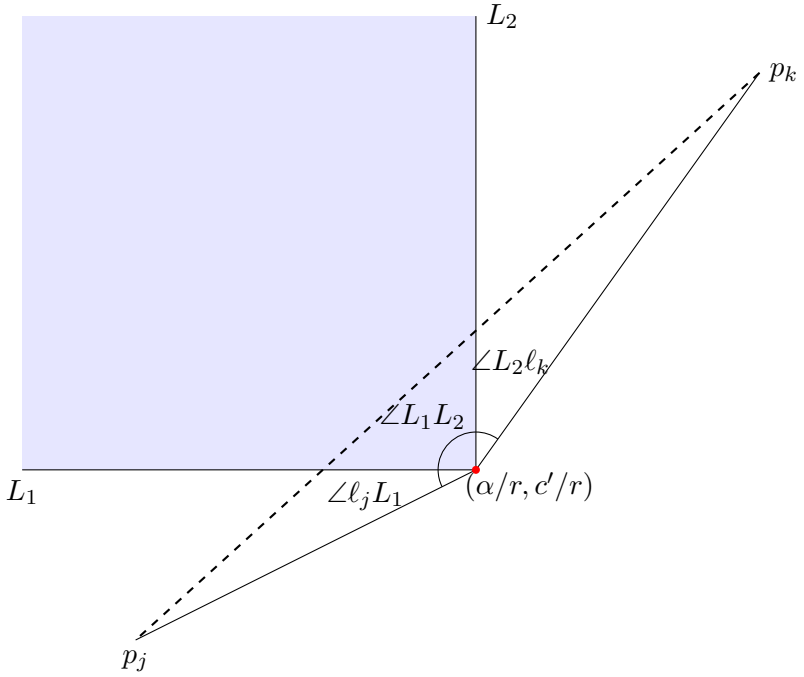


Figure 2: Illustration of $\mathcal{G}$ with the points p_j, p_k and the angles $\angle \ell_j L_1, \angle L_1 L_2, \angle L_2 \ell_k$ drawn in. Clearly the line $\overline{p_j p_k}$ only crosses $\mathcal{G}$ when the total angle is less than π.

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 $(\alpha/r, c'/r)$ to p_j and p_k, respectively. Then, as can be seen in Figure 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. 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. □

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 $\tilde{a}_j$. We do so because the trace estimation procedure of Corollary 14 provides an oracle of this form.

Lemma 16 (General version)**.** *Assume that we are given an oracle that acts as*

$$|j\rangle|0\rangle|0\rangle \mapsto |j\rangle \sum_i \beta_j^i |\tilde{a}_j^i\rangle|\psi_j^i\rangle$$

where each $|\tilde{a}_j^i\rangle$ is an approximation of a_j and the amplitudes β_j^i are such that measuring the second register with high probability returns an $\tilde{a}_j^i$ which is θ-close to a_j.

There is a quantum algorithm that uses $\widetilde{\mathcal{O}}(\sqrt{m})$ calls to the oracle described above, and the same order of two-qubit gates, and (with high probability) has the following guarantees.

- *If $\mathcal{P}_0(X) \cap \{y \in \mathbb{R}^m \colon \sum_j y_j \leq r\}$ is non-empty, then the algorithm returns a 2-sparse vector in $\mathcal{P}_{4Rr\theta}(X) \cap \{y \in \mathbb{R}^m \colon \sum_j y_j \leq r\}$.*
- *If $\mathcal{P}_{4Rr\theta}(X) \cap \{y \in \mathbb{R}^m \colon \sum_j y_j \leq r\}$ is empty, then the algorithm correctly concludes that $\mathcal{P}_0(X) \cap \{y \in \mathbb{R}^m \colon \sum_j y_j \leq r\}$ is empty.*

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

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 $(\alpha/r, (\tilde{c}-\theta)/r - \theta)$ and the lines L_1, L_2 (see Figure 2). We now apply our generalized minimum-finding algorithm with runtime $\widetilde{\mathcal{O}}(\sqrt{m})$ (see Theorem 49) starting with a uniform superposition over the js to 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 similar for ℓ and L_2). From this point on we can simply consider the model in the simple version of this lemma, 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.

It follows from Lemma 6 and Lemma 15 that 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 $(\tilde{a}_\ell, b_\ell)$ and $(\tilde{a}_k, b_k)$ lies in $\mathcal{G}$. On the other hand, if $\mathcal{P}_{4Rr\theta}(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$ is empty, then the same lemmas guarantee that we correctly conclude that $\mathcal{P}_0(X) \cap \{y \in \mathbb{R}^m : \sum_j y_j \leq r\}$ is empty. □

2.4 Total runtime

We are now ready to add our quantum implementations of the trace calculations and the oracle to the classical Arora-Kale framework.

Theorem 1. *Instantiating Meta-Algorithm 1 using the trace calculation algorithm from Section 2.2 and the oracle from Section 2.3 (with width-bound $w := r + 1$), and using this to do a binary search for* OPT $\in [-R, R]$ *(using different guesses α for* OPT*), gives a quantum algorithm for solving SDPs of the form (1), which (with high probability) produces a feasible solution y to the dual program which is optimal up to an additive error ε, and uses*

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

queries to the input matrices and the same order of other gates.

Proof. Using our implementations of the different building blocks, it remains to calculate what the total complexity will be when they are used together.

Cost of the oracle for $H^{(t)}$. The first problem in each iteration is to obtain access to an oracle for $H^{(t)}$. In each iteration the oracle will produce a $y^{(t)}$ that is at most 2-sparse, and hence in the $(t+1)$th iteration, $H^{(t)}$ is a linear combination of $2t$ of the A_j matrices and the C matrix.

We can write down a sparse representation of the coefficients of the linear combination that gives $H^{(t)}$ in each iteration by adding the new terms coming from $y^{(t)}$. This will clearly not take longer than $\widetilde{\mathcal{O}}(T)$, since there are only a constant number of terms to add for our oracle. As we will see, this term will not dominate the complexity of the full algorithm.

Using such a sparse representation of the coefficients, one query to a sparse representation of $H^{(t)}$ will cost $\widetilde{\mathcal{O}}(st)$ queries to the input matrices and $\widetilde{\mathcal{O}}(st)$ other gates. For a detailed explanation and a matching lower bound, see Appendix D.

Cost of the oracle for $\mathrm{Tr}(A_j\rho)$. In each iteration $M^{(t)}$ is made to have operator norm at most 1. This means that

$$\left\|-\eta H^{(t)}\right\| \leq \eta \sum_{\tau=1}^{t} \left\|M^{(\tau)}\right\| \leq \eta t.$$

Furthermore we know that $H^{(t)}$ is at most $d := s(2t+1)$-sparse. Calculating $\mathrm{Tr}(A_j\rho)$ for one index j up to an additive error of $\theta := \varepsilon/(12Rr)$ can be done using the algorithm from Corollary 14. This will take

$$\widetilde{\mathcal{O}}\left(\sqrt{n}\frac{\|H\|d}{\theta}\right) = \widetilde{\mathcal{O}}\left(\sqrt{n}s\frac{\eta t^2 Rr}{\varepsilon}\right)$$

queries to the oracle for $H^{(t)}$ and the same order of other gates. Since each query to $H^{(t)}$ takes $\widetilde{\mathcal{O}}(st)$ queries to the input matrices, this means that

$$\widetilde{\mathcal{O}}\left(\sqrt{n}s^2\frac{\eta t^3 Rr}{\varepsilon}\right)$$

queries to the input matrices will be made, and the same order of other gates, for each approximation of a $\mathrm{Tr}(A_j\rho)$ (and similarly for approximating $\mathrm{Tr}(C\rho)$).

Total cost of one iteration. Lemma 16 tells us that we will use $\widetilde{\mathcal{O}}(\sqrt{m})$ calculations of $\mathrm{Tr}(A_j\rho)$, and the same order of other gates, to calculate a classical description of a 2-sparse $y^{(t)}$. This brings the total cost of one iteration to

$$\widetilde{\mathcal{O}}\left(\sqrt{nm}s^2\frac{\eta t^3 Rr}{\varepsilon}\right)$$

queries to the input matrices, and the same order of other gates.

Total quantum runtime for SDPs. Since $w \leq r+1$ we can set $T = \widetilde{\mathcal{O}}\left(\frac{R^2 r^2}{\varepsilon^2}\right)$. With $\eta = \sqrt{\frac{\ln(n)}{T}}$, summing over all iterations in one run of the algorithm gives a total cost of

$$\widetilde{\mathcal{O}}\left(\sum_{t=1}^{T} \sqrt{nm} s^2 \frac{\eta t^3 Rr}{\varepsilon}\right) = \widetilde{\mathcal{O}}\left(\sqrt{nm} s^2 \frac{\eta T^4 Rr}{\varepsilon}\right) = \widetilde{\mathcal{O}}\left(\sqrt{nm} s^2 \left(\frac{Rr}{\varepsilon}\right)^8\right)$$

queries to the input matrices and the same order of other gates. □

Total quantum runtime for LPs. The final complexity of our algorithm contains a factor $\widetilde{\mathcal{O}}(sT)$ that comes from the sparsity of the $H^{(t)}$ matrix. This assumes that when we add the input matrices together, the rows become less sparse. This need not happen for certain SDPs. For example, in the SDP relaxation of MAXCUT, the $H^{(t)}$ will always be d-sparse, where d is the degree of the graph. A more important class of examples is that of linear programs: since LPs have diagonal A_j and C, their sparsity is $s = 1$, and even the sparsity of the $H^{(t)}$ is always 1. This, plus the fact that the traces can be computed without a factor $\|H\|$ in the complexity (as shown in Corollary 12 in Section 2.2.2), means that our algorithm solves LPs with

$$\widetilde{\mathcal{O}}\left(\sqrt{nm}\left(\frac{Rr}{\varepsilon}\right)^5\right)$$

queries to the input matrices and the same order of other gates.

Total classical runtime. Using the classical techniques for trace estimation from Appendix A, and the classical version of our oracle (Lemma 16), we are also able to give a general classical instantiation of the Arora-Kale framework. The final complexity will then be

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

The better dependence on Rr/ε and s, compared to our quantum algorithm, comes from the fact that we now have the time to write down intermediate results explicitly. For example, we do not need to recalculate parts of $H^{(t)}$ for every new query to it, instead we can just calculate it once at the start of the iteration by adding $M^{(t)}$ to $H^{(t-1)}$ and writing down the result.

Further remarks. We want to stress again that our solver is meant to work for *all* SDPs. In particular, it does not use the structure of a specific SDP. As we show in the next section, every oracle that works for all SDPs must have large width. To obtain quantum speedups for a *specific* class of SDPs, it will be necessary to develop oracles 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 doing exactly that: they develop specialized oracles for those problems.

3 Downside of this method: general oracles are restrictive

In this section we show some of the limitations of a method that uses sparse or general oracles, i.e., ones that are not optimized for the properties of specific SDPs. We will start by discussing

sparse oracles in the next section. We will use a counting argument to show that sparse solutions cannot hold too much information about a problem's solution. In Section 3.2 we will show that width-bounds that do not depend on the specific structure of an SDP are for many problems not efficient. As in the rest of the paper, we will assume the notation of Section 2, in particular of Meta-Algorithm 1.

3.1 Sparse oracles are restrictive

Lemma 17. *If, for some specific SDP of the form (1), every ε-optimal dual-feasible vector has at least ℓ non-zero elements, then the width w of any k-sparse $\mathsf{Oracle}_{\varepsilon/3}$ for this SDP is such that $\frac{Rw}{\varepsilon} = \Omega\Big(\sqrt{\frac{\ell}{k\ln(n)}}\Big)$.*

Proof. The vector $\bar{y}$ returned by Meta-Algorithm 1 is, by construction, the average of T vectors $y^{(t)}$ that are all k-sparse, plus one extra 1-sparse term of $\frac{\varepsilon}{R}e_1$, and hence $\ell \leq kT+1$. The stated bound on $\frac{Rw}{\varepsilon}$ then follows directly by combining this inequality with $T = \mathcal{O}(\frac{R^2w^2}{\varepsilon^2}\ln(n))$. □

The oracle presented in Section 2.3 always provides a 2-sparse vector y. This implies that if an SDP requires an ℓ-sparse dual solution, we must have $\frac{Rw}{\varepsilon} = \Omega(\sqrt{\ell/\ln(n)})$. This in turn means that the upper bound on the runtime of our algorithm will be of order $\ell^4\sqrt{nm}s^2$. This is clearly bad if ℓ is of the order n or m.

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.

Definition 18. *A problem is defined by a function f that, for every element p of the problem domain $\mathcal{D}$, gives a subset of the solution space $\mathcal{S}$, consisting of the solutions that are considered correct. We say a family of SDPs, $\{SDP^{(p)}\}_{p\in\mathcal{D}}$, solves the problem via the dual if there is an $\varepsilon \geq 0$ and a function g such that for every $p \in \mathcal{D}$ and every ε-optimal dual-feasible vector $y^{(p)}$ to $SDP^{(p)}$:*

$$g(y^{(p)}) \in f(p).$$

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 the coordinates of $\mathbb{R}^n$, then a dual solution should have a lot of unique permutations and hence cannot be too sparse.

Theorem 19. *Consider a problem and a family of SDPs as in Definition 18. Let $\mathcal{T} \subseteq \mathcal{D}$ be such that for all $p, q \in \mathcal{T}$:*

- *$f(p) \cap f(q) = \emptyset$. 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 $A_j^{(p)}$ be the constraints of $SDP^{(p)}$ and $A_j^{(q)}$ those from $SDP^{(q)}$ (and define $C^{(p)}$, $C^{(q)}$, $b_j^{(p)}$, and $b_j^{(q)}$ in the same manner). Then there exist $\sigma \in S_n$, $\pi \in S_m$ s.t. $\sigma^{-1} A^{(p)}_{\pi(j)} \sigma = A_j^{(q)}$ (and $\sigma^{-1} C^{(p)} \sigma = C^{(q)}$, $b^{(p)}_{\pi(j)} = b_j^{(q)}$). That is, the SDPs are the same up to permutations of the labels of the constraints and permutations of the coordinates of $\mathbb{R}^n$.*

If $y^{(p)}$ is an ε-optimal dual-feasible vector to $SDP^{(p)}$ for some $p \in \mathcal{T}$, then $y^{(p)}$ is at least $\frac{\log(|\mathcal{T}|)}{\log m}$-dense (i.e., has at least that many non-zero entries).

Proof. We first observe that, with $SDP^{(p)}$ and $SDP^{(q)}$ as in the lemma, if $y^{(p)}$ is an ε-optimal dual-feasible vector of $SDP^{(p)}$, then $y^{(q)}$ defined by

$$y_j^{(q)} := y^{(p)}_{\pi(j)} = \pi(y^{(p)})_j$$

is an ε-optimal dual vector for $SDP^{(q)}$. Here we use the fact that a permutation of the n coordinates in the primal does not affect the dual solutions. Since $f(p) \cap f(q) = \emptyset$ we know that $g(y^{(p)}) \neq g(y^{(q)})$ and so $y^{(p)} \neq y^{(q)}$. Since this is true for every q in $\mathcal{T}$, there should be at least $|\mathcal{T}|$ different vectors $y^{(q)} = \pi(y^{(p)})$.

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

$$\frac{\log |\mathcal{T}|}{\log m} \leq k.$$

□

Example. Consider the (s,t)-mincut problem, i.e., the dual of the (s,t)-maxflow. Specifically, consider a simple instance of this problem: the union of two complete graphs of size $z+1$, where s is in one subgraph and t in the other. Let the other vertices be labeled by $\{1, 2, \ldots, 2z\}$. Every assignment of the labels over the two halves gives a unique mincut, in terms of which labels 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 $\binom{2z}{z}$ problems that require a different output. So for every family of 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[7]

$$\frac{\log \binom{2z}{z}}{\log m} \geq \frac{z}{\log m},$$

where we used that $\binom{2z}{z} \geq \frac{2^{2z}}{2\sqrt{z}}$. In particular this holds for the standard linear programming formulation of the (s,t)-maxflow/(s,t)-mincut problem.

3.2 General width-bounds are restrictive for certain SDPs

In this section we will show that width-bounds can be restrictive when they do not consider the specific structure of an SDP.

[7] Here m is the number of constraints, not the number of edges in the graph.

Definition 20. *An algorithm O is called a* general oracle *if, when provided with an error parameter ε, it correctly implements an $\mathsf{Oracle}_\varepsilon$ (as in Algorithm 2) for all inputs. We use O_ε to denote the algorithm provided by O with error parameter ε fixed. A function $w(n, m, s, r, R, \varepsilon)$ is called a* general width-bound *for a general oracle if, for every $0 < \varepsilon < 1/2$, the value $w(n, m, s, r, R, \varepsilon)$ is a correct width-bound (see Definition 4) for O_ε for every SDP with parameters n, m, s, r, and R. In particular, the function w may not depend on the structure of the input $A_1, \ldots, A_m$, C, b or on the value of α.*

We will show that general width-bounds need to scale with r^* (recall that r^* denotes the smallest ℓ_1-norm of an optimal solution to the dual). We then go on to show that if two SDPs in a class can be combined to get another element of that class in a natural manner, then, under some mild conditions, r^* will be of the order n and m for some instances of the class.

We start by showing, for specifically constructed LPs, a lower bound on the width of any oracle. Although these LPs will not solve any useful problem, every general width-bound should also apply to these LPs. This gives a lower bound on general width-bounds.

Lemma 21. *For every $n \geq 3$, $m \geq 3$, $s \geq 1$, $R^* \geq 1$, $r^* > 0$, there is an SDP with these parameters such that for any $0 \leq \varepsilon \leq 1/2$ any $\mathsf{Oracle}_\varepsilon$ for this SDP has width at least $r^*/2$.*

Proof. We will construct an LP for $n = m = 3$. This is enough to prove the lemma since LPs are a subclass of SDPs and we can increase n, m, and s by adding more dimensions and s-dense SDP constraints that do not influence the analysis below. For some $k > 0$, consider the following LP

$$\begin{aligned} \max \quad & (1,0,0)x \\ \text{s.t.} \quad & \begin{bmatrix} 1 & 1 & 1 \\ 1/k & 1 & 0 \\ -1 & 0 & -1 \end{bmatrix} x \leq \begin{bmatrix} R \\ 0 \\ -R \end{bmatrix} \\ & x \geq 0 \end{aligned}$$

where the first row is the primal trace constraint. Notice that $x_1 = x_2 = 0$ due to the second constraint. This implies that OPT $= 0$ and, due to the last constraint, that $x_3 \geq R$. Notice that $(0, 0, R)$ is actually an optimal solution, so $R^* = R$.

To calculate r^*, look at the dual of the LP:

$$\begin{aligned} \min \quad & (R,0,-R)y \\ \text{s.t.} \quad & \begin{bmatrix} 1 & 1/k & -1 \\ 1 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix} y \geq \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ & y \geq 0, \end{aligned}$$

due to strong duality its optimal value is 0 as well. This implies $y_1 = y_3$, so the first constraint becomes $y_2 \geq k$. This in turn implies $r^* \geq k$, which is actually attained (by $y = (0, k, 0)$) so $r^* = k$.

Since the oracle and width-bound should work for every $x \in \mathbb{R}^3_+$ and every α, they should in particular work for $x = (R, 0, 0)$ and $\alpha = 0$. In this case the polytope for the oracle becomes

$$\begin{aligned} \mathcal{P}_\varepsilon(x) := \{y \in \mathbb{R}^m : \; & y_1 - y_3 \leq 0, \\ & y_1 - y_3 + y_2/k \geq 1 - \varepsilon/R, \\ & y \geq 0\}. \end{aligned}$$

since $b^T y = y_1 - y_3$, $c^T x = 1$, $a_1^T x = 1$, $a_2^T x = 1/k$ and $a_3^T x = -1$. This implies that for every $y \in \mathcal{P}_\varepsilon(x)$, we have $y_2 \geq k(1 - \varepsilon/R) \geq k/2 = r^*/2$, where the second inequality follows from the assumptions that $\varepsilon \leq 1/2$ and $R \geq 1$.

Notice that the term

$$\left\| \sum_{j=1}^{m} y_j A_j - C \right\|$$

in the definition of width for an SDP becomes

$$\left\| A^T y - c \right\|_\infty$$

in the case of an LP. In our case, due to the second constraint in the dual, we know that

$$\left\| A^T y - c \right\|_\infty \geq y_1 + y_2 \geq \frac{r^*}{2}$$

for every vector y from $\mathcal{P}_\varepsilon(x)$. This shows that any oracle has width at least $r^*/2$ for this LP. □

Corollary 22. *For every general width-bound $w(n, m, s, r, R, \varepsilon)$, if $n, m \geq 3$, $s \geq 1$, $r > 0$, $R > 1$, and $\varepsilon \leq 1/2$, then*

$$w(n, m, s, r, R, \varepsilon) \geq \frac{r}{2}.$$

Proof. Consider the LP given by Lemma 21 with $r^* = r$. Using a general oracle with general width-bound w for this LP implies the corollary. □

Note that this bound applies to both our algorithm and the one given by Brandão and Svore. It turns out that for many classes of SDPs it is natural to assume that m, r^*, and R^* grow linearly with n, and that the "logical" choice of ε also scales linearly with n. We now argue that this is for instance the case when SDPs in a class combine in a natural manner. As an example, consider the MAXCUT problem. For, e.g., d-regular graphs the MAXCUT value grows linearly with the number of vertices n. Therefore, one is generally interested in finding a constant *multiplicative* approximation to the optimal value. For d-regular graphs this would thus translate to an *additive* error which scales linearly with the number of vertices. We argue below that for the SDP-relaxation it is also natural to assume that r and R grow linearly in n. 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):

$$\begin{aligned} \max \quad & \mathrm{Tr}\left(L(G^{(1)})X^{(1)}\right) \\ \text{s.t.} \quad & \mathrm{Tr}\left(X^{(1)}\right) \leq n^{(1)} \\ & \mathrm{Tr}\left(E_{jj}X^{(1)}\right) \leq 1 \text{ for } j = 1, \ldots, n^{(1)} \\ & X^{(1)} \succeq 0 \end{aligned} \qquad \begin{aligned} \max \quad & \mathrm{Tr}\left(L(G^{(2)})X^{(2)}\right) \\ \text{s.t.} \quad & \mathrm{Tr}\left(X^{(2)}\right) \leq n^{(2)} \\ & \mathrm{Tr}\left(E_{jj}X^{(2)}\right) \leq 1 \text{ for } j = 1, \ldots, n^{(2)} \\ & X^{(2)} \succeq 0 \end{aligned}$$

Where $L(G)$ is the Laplacian of a graph. 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:

$$\begin{aligned} \max \quad & \mathrm{Tr}\Big(L(G^{(1)})X^{(1)}\Big) + \mathrm{Tr}\Big(L(G^{(2)})X^{(2)}\Big) \\ \text{s.t.} \quad & \mathrm{Tr}\Big(X^{(1)}\Big) + \mathrm{Tr}\Big(X^{(2)}\Big) \leq n^{(1)} + n^{(2)} \\ & \mathrm{Tr}\Big(E_{jj}X^{(1)}\Big) \leq 1 \text{ for } j = 1, \ldots, n^{(1)} \\ & \mathrm{Tr}\Big(E_{jj}X^{(2)}\Big) \leq 1 \text{ for } j = 1, \ldots, n^{(2)} \\ & X^{(1)}, X^{(2)} \succeq 0. \end{aligned}$$

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 $n^{(1)} + n^{(2)} = R^{*(1)} + R^{*(2)}$. Likewise, it is natural to assume that the desired approximation error for the combined SDP is the sum of the desired errors for the individual SDPs: starting with feasible solutions $X^{(i)}$ that are $\varepsilon^{(i)}$-approximate solutions to the two SDP-relaxations ($i = 1, 2$), the matrix $X^{(1)} \oplus X^{(2)}$ is an $(\varepsilon^{(1)} + \varepsilon^{(2)})$-approximate solution to the combined SDP. It remains to see what happens to r^*, and so, for general width-bounds, what happens to w. As we will see later in this section, under some mild conditions, these kind of combinations imply that there are MAXCUT-relaxation SDPs for which r^* also increases linearly, but this requires a bit more work.

Definition 23. *We say that a class of SDPs (each with an associated allowed approximation error) is* combinable *if there is a $k \geq 0$ so that 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* $j \in [k]$.
- $A_j^{(c)} = A_j^{(a)} \oplus \mathbf{0}$ *and* $b_j^{(c)} = b_j^{(a)}$ *for* $j = k+1, \ldots, m^{(a)}$.
- $A^{(c)}_{m^{(a)}+j-k} = \mathbf{0} \oplus A_j^{(b)}$ *and* $b^{(c)}_{m^{(a)}+j-k} = b_j^{(b)}$ *for* $j = k+1, \ldots, m^{(b)}$.
- $\varepsilon^{(c)} = \varepsilon^{(a)} + \varepsilon^{(b)}$.

In other words, some fixed set of constraints are summed pairwise, and the remaining constraints get added separately.

The motivation for the above definition reflects the following: if $X^{(a)}$ and $X^{(b)}$ are feasible solutions to $SDP^{(a)}$ and $SDP^{(b)}$ that are $\varepsilon^{(a)}$-approximate and $\varepsilon^{(b)}$-approximate solutions respectively, then $X^{(a)} \oplus X^{(b)}$ is an $(\varepsilon^{(a)} + \varepsilon^{(b)})$-approximate solution to $SDP^{(c)}$.

Furthermore, note that this is a natural generalization of the combining property of the MAXCUT relaxations (in that case $k = 1$ to account for the constraint giving the trace bound).

Theorem 24. *If a class of SDPs is combinable and there is an element $SDP^{(1)}$ for which every optimal dual solution has the property that*

$$\sum_{j=k+1}^{m} y_m \geq \delta$$

for some $\delta > 0$, then there is a sequence $(SDP^{(t)}, \varepsilon^{(t)})_{t\in\mathbb{N}}$ in the class such that $\frac{R^{*(t)}r^{*(t)}}{\varepsilon^{(t)}}$ increases linearly in $n^{(t)}$, $m^{(t)}$ and t, and $SDP^{(t)}$ is the t-fold combination of $SDP^{(1)}$ with itself.

Proof. The sequence we will consider is the t-fold combination of $SDP^{(1)}$ with itself. If $SDP^{(1)}$ is

$$\begin{aligned} \max \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \le b_j \quad \text{for } j \in [m^{(1)}], \\ & X \succeq 0 \end{aligned} \qquad\qquad \begin{aligned} \min \quad & \sum_{j=1}^{m^{(1)}} b_j y_j \\ \text{s.t.} \quad & \sum_{j=1}^{m^{(1)}} y_j A_j - C \succeq 0, \\ & y \ge 0 \end{aligned}$$

then $SDP^{(t)}$ is

$$\begin{aligned} \max \quad & \sum_{i=1}^{t} \mathrm{Tr}(CX_i) \\ \text{s.t.} \quad & \sum_{i=1}^{t} \mathrm{Tr}(A_j X_i) \le t b_j && \text{for } j \in [k], \\ & \mathrm{Tr}(A_j X_i) \le b_j && \text{for } j = k+1, \dots, m^{(1)} \text{ and } i = 1, \dots, t \\ & X_i \succeq 0 && \text{for all } i = 1, \dots, t \end{aligned}$$

with dual

$$\begin{aligned} \min \quad & \sum_{j=1}^{k} t b_j y_j + \sum_{i=1}^{t} \sum_{j=k+1}^{m^{(1)}} b_j y_j^i \\ \text{s.t.} \quad & \sum_{j=1}^{k} y_j A_j + \sum_{j=k+1}^{m^{(1)}} y_j^i A_j \succeq C \text{ for } i = 1, \dots, t \\ & y, y^i \ge 0. \end{aligned}$$

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 $\mathrm{OPT}^{(t)} \ge t \cdot \mathrm{OPT}^{(1)}$. Furthermore, let $y^{(1)}$ be an optimal dual solution of $SDP^{(1)}$, then $(y_1^{(1)}, \dots, y_k^{(1)}) \oplus \left(y_{k+1}^{(1)}, \cdots, y_{m^{(1)}}^{(1)}\right)^{\oplus t}$ is a feasible dual solution for $SDP^{(t)}$ with objective value $t \cdot \mathrm{OPT}^{(1)}$, so $\mathrm{OPT}^{(t)} = t \cdot \mathrm{OPT}^{(1)}$.

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 $b^T(\tilde{y} \oplus y^i) \ge \mathrm{OPT}^{(1)}$. 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 $\left\|(\tilde{y} \oplus y^i)\right\|_1 \ge r^{*(1)}$ by definition and $\left\|y^i\right\|_1 \ge \delta$. We conclude that $r^{*(t)} \ge r^{*(1)} - \delta + t\delta$.

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 23 we take $k = n^{(1)}$, however, contrary to that definition, we even sum the psd constraints:

$$\begin{aligned} \max \quad & \sum_{i=1}^{t} \mathrm{Tr}(CX_i) \\ \text{s.t.} \quad & \sum_{i=1}^{t} \mathrm{Tr}(A_j X_i) \leq t b_j \quad \text{for } j \in [m^{(1)}], \\ & \sum_{i=1}^{t} X_i \succeq 0. \end{aligned}$$

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, and hence has trace at least $t \cdot R^{*(1)}$, so $R^{*(t)} \geq t \cdot R^{*(1)}$.

We conclude that

$$\frac{R^{*(t)} r^{*(t)}}{\varepsilon^{(t)}} \geq \frac{t R^{*(1)} (r^{*(1)} + (t-1)\delta)}{t\varepsilon^{(1)}} = \Omega(t)$$

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, $R^* r^* / \varepsilon$ increases linearly in n and m for some instances. Hence, using $R^* \leq R$ and Lemma 21, Rw/ε grows at least linearly when a general width-bound is used.

4 Lower bounds on the quantum query complexity

In this section we will show that every LP-solver (and hence every SDP-solver) that can distinguish two optimal values with high probability needs $\Omega\Big(\sqrt{\max\{n,m\}}(\min\{n,m\})^{3/2}\Big)$ quantum queries in the worst case.

For the lower bound on LP-solving we will give a reduction from a composition of Majority and OR functions.

Definition 25. *Given input bits $Z_{ij\ell} \in \{0,1\}^{a \times b \times c}$ the problem of calculating*

$$\begin{aligned} MAJ_a(& \\ & OR_b(MAJ_c(Z_{111}, \ldots, Z_{11c}), \ldots, MAJ_c(Z_{1b1}, \ldots, Z_{1bc})), \\ & \ldots, \\ & OR_b(MAJ_c(Z_{a11}, \ldots, Z_{a1c}), \ldots, MAJ_c(Z_{ab1}, \ldots, Z_{abc})) \\ ) & \end{aligned}$$

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 calculations, then* $|\tilde{Z}| \in \{a/2, a/2+1\}$.

is called the promise MAJ_a-OR_b-MAJ_c *problem.*

Lemma 26. *It takes at least* $\Omega(a\sqrt{b}\,c)$ *queries to the input to solve the promise* MAJ_a-OR_b-MAJ_c *problem.*

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, Lemma A.3 (Lemma 6 in the arXiv version)] showed that this even holds for promise functions. □

Lemma 27. *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 $i \in [a], j \in [b]$. This implies that

- If the ith OR is 0, then all of its inner MAJ functions are 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 is 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

$$\sum_{i=1}^{a} \max_{j\in[b]} \sum_{\ell=1}^{c} Z_{ij\ell} = a\frac{c}{2} + |\tilde{Z}|.$$

Hence determining the left-hand side will determine $|\tilde{Z}|$; 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 28. *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 to an entry of the input matrix or vector costs at most 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:

$$\begin{aligned} \mathrm{OPT} = \max \quad & \sum_{k=1}^{c} w_k \\ \text{s.t.} \quad & \begin{bmatrix} I & -Z^1 & \cdots & -Z^a \\ 0 & \mathbf{1}^T & & \\ 0 & & \ddots & \\ 0 & & & \mathbf{1}^T \end{bmatrix} \begin{bmatrix} w \\ v^{(1)} \\ \vdots \\ v^{(a)} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\ & v^1, \ldots, v^a \in \mathbb{R}^b_+, w \in \mathbb{R}^c_+. \end{aligned}$$

Notice 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

$$\sum_{j=1}^{b} v_j^{(i)} \leq 1.$$

The constraints involving w say that for every $k \in [c]$

$$w_k \leq \sum_{i=1}^{a} \sum_{j=1}^{b} v_j^{(i)} Z_{kj}^{(i)} = \sum_{i=1}^{a} (Z^{(i)} v^{(i)})_k,$$

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. 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 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 $\|Z^{(i)}v^{(i)}\|_1 = \max_{j \in [b]} \sum_{\ell=1}^{c} Z_{\ell j}^{(i)}$. Putting everything together gives:

$$\text{OPT} = \sum_{i=1}^{a} \left\| Z^{(i)} v^{(i)} \right\|_1 = \sum_{i=1}^{a} \max_{j \in [b]} \sum_{\ell=1}^{c} Z_{\ell j}^{(i)} = \sum_{i=1}^{a} \max_{j \in [b]} \sum_{\ell=1}^{c} Z_{ij\ell}.$$

□

Theorem 29. *There is a family of LPs, with $m \leq n$ and two possible integer optimal values, that require at least $\Omega(\sqrt{n}\, m^{3/2})$ quantum queries to the input to distinguish those two values.*

Proof. Let $a = c = m/2$ and $b = \frac{n-c}{a} = \frac{2n}{m} - 1$, so that $n = c + ab$ and $m = c + a$. By Lemma 28 there exists an LP with $n = c + ab$ and $m = c + a$ that calculates

$$\sum_{i=1}^{a} \max_{j \in [b]} \sum_{\ell=1}^{c} Z_{ij\ell}$$

for an input Z to the promise MAJ_a-OR_b-MAJ_c problem. By Lemma 27, calculating this value will solve the promise MAJ_a-OR_b-MAJ_c problem. By Lemma 26 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. □

Corollary 30. *Distinguishing two optimal values of an LP (and hence also of an SDP) with additive error $\varepsilon < 1/2$ requires*

$$\Omega\left(\sqrt{\max\{n,m\}}(\min\{n,m\})^{3/2}\right)$$

quantum queries to the input matrices in the worst case.

Proof. This follows from Theorem 29 and LP duality. □

It is important to note that the parameters R and r from the Arora-Kale algorithm are not constant in this family of LPs ($R, r = \Theta(\min\{n,m\}^2)$ here), and hence this lower bound does not contradict the scaling with $\sqrt{mn}$ of the complexity of our SDP-solver or Brandão and Svore's. Since we show in the appendix that one can always rewrite the LP (or SDP) so that 2 of the parameters R, r, ε are constant, the lower bound implies that any algorithm with a sublinear dependence on m or n has to depend at least polynomially on Rr/ε. For example, the above family of LPs shows that an algorithm with a $\sqrt{mn}$ dependence has to have an $(Rr/\varepsilon)^\kappa$ factor in its complexity with $\kappa \geq 1/4$. It remains an open question whether a lower bound of $\Omega(\sqrt{mn})$ can be proven for a family of LPs/SDPs where ε, R and r all constant.

5 Conclusion

In this paper we gave better algorithms and lower bounds for quantum SDP-solvers, improving upon recent work of Brandão and Svore [BS17]. Here are a few directions for future work:

- **Better upper bounds.** The runtime of our algorithm, like the earlier algorithm of Brandão and Svore, has better dependence on m and n than the best classical SDP-solvers, but worse dependence on s and on Rr/ε. In subsequent work (see the introduction), these dependencies have been improved, but there is room for further improvement.

- **Applications of our algorithm.** As mentioned, both our and Brandão-Svore's quantum SDP-solvers only improve upon the best classical algorithms for a specific regime of parameters, namely where $mn \gg Rr/\varepsilon$. Unfortunately, we don't know particularly interesting problems in combinatorial optimization in this regime. As shown in Section 3, many natural SDP formulations will not fall into this regime. However, it would be interesting to find useful SDPs for which our algorithm gives a significant speed-up.

- **New algorithms.** As in the work by Arora and Kale, it might be more promising to look at oracles (now quantum) that are designed for specific SDPs. Such oracles could build on the techniques developed here, or develop totally new techniques. It might also be possible to speed up other classical SDP solvers, for example those based on interior-point methods.

- **Better lower bounds.** Our lower bounds are probably not optimal, particularly for the case where m and n are not of the same order. The most interesting case would be to get lower bounds that are simultaneously tight in the parameters m, n, s, and Rr/ε.

Acknowledgments. An earlier version of this paper appeared in the proceedings of the FOCS'17 conference [vAGGdW17].

We thank Fernando Brandão for sending us several drafts of [BS17] and for answering our many questions about their algorithms. We thank Stacey Jeffery for pointing us to [Kim13], and Andris Ambainis and Robin Kothari for useful discussions and comments. Thanks to Monique Laurent for finding a bug in an earlier version, which we fixed. We also thank the anonymous referees of FOCS'17 and Quantum for very helpful comments that improved the presentation.

JvA and SG are supported by the Netherlands Organization for Scientific Research (NWO), grant number 617.001.351. AG and RdW are supported by ERC Consolidator Grant 615307-QPROGRESS. RdW is also partially supported by NWO through Gravitation-grant Quantum Software Consortium - 024.003.037, and through QuantERA project QuantAlgo 680-91-034.

References

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. 4, 11

[AK16] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *Journal of the ACM*, 63(2):12:1–12:35, 2016. Earlier version in STOC'07. 4, 11, 12, 13, 42

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007. Earlier version in FOCS'04. arXiv: quant-ph/0311001 5

[vAG19a] Joran van Apeldoorn and András Gilyén. Improvements in quantum SDP-solving with applications. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 99:1–99:15, 2019. arXiv: 1804.05058 9

[vAG19b] Joran van Apeldoorn and András Gilyén. Quantum algorithms for zero-sum games. arXiv: 1904.03180, 2019. 9

[vAGGdW17] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-solvers: Better upper and lower bounds. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 403–414, 2017. arXiv: 1705.01843 39

[vAGGdW20] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020. arXiv: 1809.00643 9

[AS08] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley-Interscience, third edition, 2008. 49

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. arXiv: quant-ph/9605034 56, 60, 62

[BCC+15] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical Review Letters*, 114(9):090502, 2015. arXiv: 1412.4687 8, 46, 50

[BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 792–809, 2015. arXiv: 1501.01715 8, 11, 46, 47, 50

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *Contemporary Mathematics Series*, pages 53–74. AMS, 2002. arXiv: quant-ph/0005055 17, 58

[BKL+19] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi 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)*, pages 27:1–27:14, 2019. arXiv: 1710.02581 9

[BS17] Fernando G. S. L. Brandão and Krysta M. Svore. Quantum speed-ups for solving semidefinite programs. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 415–426, 2017. arXiv: 1609.05537 1, 6, 15, 38, 39

[CCLW20] Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:221, 2020. arXiv: 1809.01731 9

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society A*, 454(1969):339–354, 1998. arXiv: quant-ph/9708016 63

[CKS17] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017. arXiv: 1511.02306 8, 46, 47, 58

[CS17] Anirban Narayan Chowdhury and Rolando D. Somma. Quantum algorithms for Gibbs sampling and hitting-time estimation. *Quantum Information and Computation*, 17(1&2):41–64, 2017. arXiv: 1603.02940 6, 20, 46, 56

[CW12] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information and Computation*, 12(11&12):901–924, 2012. arXiv: 1202.5822 8, 46, 50

[Dan51] George B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In *Activity Analysis of Production and Allocation*, Cowles Commission Monograph No. 13, pages 339–347. John Wiley & Sons Inc., New York, N. Y., 1951. 4

[DH96] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. arXiv: quant-ph/9607014, 1996. 7, 8, 19, 58, 59, 63

[DHHM06] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. Earlier version in ICALP'04. arXiv: quant-ph/0401091 5

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. 4

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988. 3

[GR02] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. arXiv: quant-ph/0208112, 2002. 51

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing (STOC)*, pages 212–219, 1996. arXiv: quant-ph/9605043 5, 7

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*, pages 193–204, 2019. arXiv: 1806.01838 9

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. Earlier version in STOC'94. 3

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv: 0811.3171 5, 58

[Kim13] Shelby Kimmel. Quantum adversary (upper) bound. *Chicago Journal of Theoretical Computer Science*, 2013:4:1–14, 2013. Earlier version in ICALP'12. arXiv: 1101.0797 36, 39

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007. Earlier version in FOCS'04. 3

[KP18] Iordanis Kerenidis and Anupam Prakash. A quantum interior point method for LPs and SDPs. arXiv: 1808.09266, 2018. 9

[LC17] Guang Hao Low and Isaac L. Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1):010501, 2017. arXiv: 1606.02685 47

[LC19] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019. arXiv: 1610.06546 47

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai 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)*, pages 1049–1065, 2015. arXiv: 1508.04874 4, 5

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000. 63

[NN94] Y. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), 1994. 4

[PW09a] David Poulin and Pawel Wocjan. Preparing ground states of quantum many-body systems on a quantum computer. *Physical Review Letters*, 102(13):130503, 2009. arXiv: 0809.2705 8, 63

[PW09b] David Poulin and Pawel Wocjan. Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer. *Physical Review Letters*, 103(22):220502, 2009. arXiv: 0905.2199 6

[Ren16] James Renegar. "Efficient" subgradient methods for general convex optimization. *SIAM Journal on Computing*, 26(4):2649–2676, 2016. arXiv: 1605.08712 4

[Ren19] James Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1):1–35, 2019. arXiv: 1512.07569 4

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. 22

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Earlier version in FOCS'94. arXiv: quant-ph/9508027 5

[TRW05] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005. Earlier version in NIPS'04. 4

[WK12] Manfred K. Warmuth and Dima Kuzmin. Online variance minimization. *Machine Learning*, 87(1):1–32, 2012. Earlier version in COLT'06. 4

A Classical estimation of the expectation value $\mathrm{Tr}(A\rho)$

To provide contrast to Section 2.2, here we describe a *classical* procedure to efficiently estimate $\mathrm{Tr}(A\rho)$ where A is a Hermitian matrix such that $\|A\| \leq 1$, and $\rho = \exp(-H)/\mathrm{Tr}(\exp(-H))$ for some Hermitian matrix H. The results in this section can be seen as a generalization of [AK16, Section 7]. The key observation is that if we are given a Hermitian matrix $B \succeq 0$, and if we take a random vector $u = (u_1, \ldots, u_n)$ where $u_i \in \{\pm 1\}$ is uniformly distributed, then, using $\mathbb{E}[u_i] = 0$, $\mathbb{E}[u_i^2] = 1$, we have

$$\begin{aligned}\mathbb{E}\Big[u^T\sqrt{B}A\sqrt{B}u\Big] &= \mathbb{E}\Big[\mathrm{Tr}\Big(\sqrt{B}A\sqrt{B}uu^T\Big)\Big] = \mathrm{Tr}\Big(\sqrt{B}A\sqrt{B}\mathbb{E}[uu^T]\Big)\\ &= \mathrm{Tr}\Big(\sqrt{B}A\sqrt{B}I\Big) = \mathrm{Tr}(AB).\end{aligned}$$

We now show that $u^T\sqrt{B}A\sqrt{B}u$ is highly concentrated around its mean by Chebyshev's inequality.

Lemma 31. *Given a Hermitian matrix A, with $||A|| \leq 1$, a psd matrix B, and a parameter $0 < \theta \leq 1$. With probability $1 - 1/16$, the average of $k = \mathcal{O}(1/\theta^2)$ independent samples from the distribution $u^T\sqrt{B}A\sqrt{B}u$ is at most $\theta\mathrm{Tr}(B)$ away from $\mathrm{Tr}(AB)$. Here $u = (u_i)$ and each $u_i \in \{\pm 1\}$ is i.i.d. uniformly distributed.*

Proof. We let F_k be the random variable $\frac{1}{k}\sum_{i=1}^k (u^{(i)})^T\sqrt{B}A\sqrt{B}u^{(i)}$, where each of the vectors $u^{(i)} \in \{\pm 1\}^n$ is sampled from the distribution described above. By the above it is clear that $\mathbb{E}[F_k] = \mathrm{Tr}(AB)$.

We will use Chebyshev's inequality which, in our setting, states that for every $t > 0$

$$\Pr(|F_k - \mathrm{Tr}(AB)| \geq t\sigma_k) \leq \frac{1}{t^2}, \tag{12}$$

here σ_k^2 is the variance of F_k. We will now upper bound the variance of F_k. First note that $\mathrm{var}(F_k) = \frac{1}{k}\mathrm{var}(u^T\sqrt{B}A\sqrt{B}u)$. It therefore suffices to upper bound the variance σ^2 of

$u^T\sqrt{B}A\sqrt{B}u$. We first write

$$\begin{aligned}\sigma^2 = \mathrm{var}(u^T\sqrt{B}A\sqrt{B}u) &= \mathbb{E}\Big[(u^T\sqrt{B}A\sqrt{B}u)^2\Big] - \mathbb{E}\Big[u^T\sqrt{B}A\sqrt{B}u\Big]^2 \\ &= \underbrace{\mathbb{E}\left[\sum_{i,j,k,l=1}^{n} u_iu_ju_ku_l(\sqrt{B}A\sqrt{B})_{ij}(\sqrt{B}A\sqrt{B})_{kl}\right]}_{(*)} - \mathrm{Tr}(\sqrt{B}A\sqrt{B})^2.\end{aligned}$$

We then calculate $(*)$ using $\mathbb{E}[u_i] = 0$, $\mathbb{E}[u_i^2] = 1$, and the independence of the u_i's:

$$\begin{aligned}(*) &= \sum_{i\neq j}(\sqrt{B}A\sqrt{B})_{ij}\Big((\sqrt{B}A\sqrt{B})_{ij} + (\sqrt{B}A\sqrt{B})_{ji}\Big) + \sum_{i,k=1}^{n}(\sqrt{B}A\sqrt{B})_{ii}(\sqrt{B}A\sqrt{B})_{kk} \\ &= \sum_{i\neq j} 2(\sqrt{B}A\sqrt{B})_{ij}^2 + \mathrm{Tr}(\sqrt{B}A\sqrt{B})^2.\end{aligned}$$

Therefore, using Cauchy-Schwarz, we have

$$\begin{aligned}\sigma^2 &= (*) - \mathrm{Tr}(\sqrt{B}A\sqrt{B})^2 = \sum_{i\neq j} 2(\sqrt{B}A\sqrt{B})_{ij}^2 \leq \sum_{i,j} 2(\sqrt{B}A\sqrt{B})_{ij}^2 \\ &= 2\mathrm{Tr}((\sqrt{B}A\sqrt{B})^2) = 2|\langle ABA, B\rangle| \leq 2|\langle ABA, ABA\rangle|^{1/2}|\langle B, B\rangle|^{1/2} \\ &= 2\mathrm{Tr}\Big(A^2BA^2B\Big)^{1/2}\mathrm{Tr}\Big(B^2\Big)^{1/2} \leq 2\mathrm{Tr}\Big(BA^2B\Big)^{1/2}\mathrm{Tr}\Big(B^2\Big)^{1/2} \leq 2\mathrm{Tr}\Big(B^2\Big) \leq 2\mathrm{Tr}(B)^2,\end{aligned}$$

where on the last line we use $\|A\| \leq 1$ and $\mathrm{Tr}(AY) \leq \|A\|\mathrm{Tr}(Y)$ for any $Y \succeq 0$, in particular for BA^2B and B^2.

It follows that $\sigma_k^2 \leq 2\mathrm{Tr}(B)^2/k$. Chebyshev's inequality (12) therefore shows that for $k = \lceil 32/\theta^2\rceil$ and $t = 4$,

$$\Pr(|F_k - \mathrm{Tr}(AB)| \geq \theta\mathrm{Tr}(B)) \leq \frac{1}{16}. \qquad \square$$

A simple computation shows that the success probability in the above lemma can be boosted to $1-\delta$ by picking the median of $\mathcal{O}(\log(1/\delta))$ repetitions. To show this, let $K = \lceil\log(1/\delta)\rceil$ and for each $i \in [K]$ let $(F_k)_i$ be the average of k samples of $u^T\sqrt{B}A\sqrt{B}u$. Let z_K denote the median of those K numbers. We have

$$\Pr(|z_K - \mathrm{Tr}(AB)| \geq \theta\mathrm{Tr}(B)) = \underbrace{\Pr(z_K \geq \theta\mathrm{Tr}(B) + \mathrm{Tr}(AB))}_{(*)} + \Pr(z_K \leq -\theta\mathrm{Tr}(B) + \mathrm{Tr}(AB))$$

We upper bound $(*)$:

$$\begin{aligned}(*) &\leq \sum_{I\subseteq[K]:|I|\geq K/2}\prod_{i\in I}\Pr((F_k)_i \geq \theta\mathrm{Tr}(B) + \mathrm{Tr}(AB)) \\ &\leq (|\{I \subseteq [K] : |I| \geq K/2\}|)\left(\frac{1}{16}\right)^{K/2} \\ &= 2^{K-1}\left(\frac{1}{4}\right)^K \\ &\leq \frac{1}{2}\left(\frac{1}{2}\right)^{\log(1/\delta)} = \frac{1}{2}\delta.\end{aligned}$$

Analogously, one can show that $\Pr(z_K \leq -\theta\mathrm{Tr}(B) + \mathrm{Tr}(AB)) \leq \frac{1}{2}\delta$. Hence

$$\Pr(|z_K - \mathrm{Tr}(AB)| \geq \theta\mathrm{Tr}(B)) \leq \delta.$$

This proves the following lemma:

Lemma 32. *Given a Hermitian matrix A, with $||A|| \leq 1$, a psd matrix B, and parameters $0 < \delta \leq 1/2$ and $0 < \theta \leq 1$. Using $k = \mathcal{O}\Big(\log(\frac{1}{\delta})/\theta^2\Big)$ samples from the distribution $u^T\sqrt{B}A\sqrt{B}u$, one can find an estimate of $\mathrm{Tr}(AB)$ that, with probability $1-\delta$, has additive error at most $\theta\mathrm{Tr}(B)$. Here $u = (u_i)$ and the $u_i \in \{\pm 1\}$ are i.i.d. uniformly distributed.*

Looking back at Meta-Algorithm 1, we would like to apply the above lemma to $B = \exp(-H)$.[8] Since it is expensive to compute the exponent of a matrix, it is of interest to consider samples from v^TAv, where v is an approximation of $\sqrt{B}u$.

Say $\Big\|\sqrt{B}u - v\Big\| \leq \kappa$ and, as always, $\|A\| \leq 1$. Then

$$\begin{aligned}
|u^T\sqrt{B}A\sqrt{B}u - v^TAv| &= |u^T\sqrt{B}A\sqrt{B}u - u^T\sqrt{B}Av + u^T\sqrt{B}Av - v^TAv| \\
&\leq |u^T\sqrt{B}A\sqrt{B}u - u^T\sqrt{B}Av| + |u^T\sqrt{B}Av - v^TAv| \\
&= |u^T\sqrt{B}A(\sqrt{B}u - v)| + |(\sqrt{B}u - v)^TAv| \\
&\leq \Big\|u^T\sqrt{B}A\Big\|\Big\|\sqrt{B}u - v\Big\| + \Big\|\sqrt{B}u - v\Big\|\|Av\| \\
&\leq \Big\|\sqrt{B}u\Big\|\|A\|\kappa + \kappa\|A\|\|v\| \\
&\leq \kappa(\Big\|\sqrt{B}u\Big\| + \|v\|) \\
&\leq \kappa(\Big\|\sqrt{B}u\Big\| + \Big\|\sqrt{B}u + v - \sqrt{B}u\Big\|) \\
&\leq \kappa\Big(\Big\|\sqrt{B}u\Big\| + \Big\|\sqrt{B}u\Big\| + \Big\|\sqrt{B}u - v\Big\|\Big) \\
&\leq 2\kappa\Big\|\sqrt{B}u\Big\| + \kappa^2
\end{aligned}$$

Now observe that we are interested in

$$\frac{\mathrm{Tr}(A\exp(-H))}{\mathrm{Tr}(\exp(-H))} = \frac{\mathrm{Tr}(A\exp(-H+\gamma I))}{\mathrm{Tr}(\exp(-H+\gamma I))}.$$

Suppose an upper bound K on $\|H\|$ is known, then we can consider $H' = H - KI$ which satisfies $H' \preceq 0$. It follows that $\|\exp(-H')\| \geq 1$ and, with $B = \exp(-H')$, therefore $\|\sqrt{B}\| \leq \|B\| \leq \mathrm{Tr}(B)$. Hence, taking $\kappa \leq \min\{\theta/\|u\|, \|\sqrt{B}u\|\} = \theta/\|u\|$,[9] we find

$$|u^T\sqrt{B}A\sqrt{B}u - v^TAv| \leq 2\kappa\Big\|\sqrt{B}u\Big\| + \kappa^2 \leq 3\kappa\Big\|\sqrt{B}u\Big\| \leq 3\kappa\Big\|\sqrt{B}\Big\|\|u\| \leq 3\theta\|B\| \leq 3\theta\mathrm{Tr}(B).$$

This shows that the additional error incurred by sampling from v^TAv is proportional to $\theta\mathrm{Tr}(B)$. Finally, a κ-approximation of $\sqrt{B}u$, with $\kappa = \theta/\|u\|$ can be obtained by using the truncated Taylor series of $\exp(-H'/2)$ of degree $p = \max\{2e\|H'\|, \log\Big(\frac{\sqrt{n}}{\theta}\Big)\}$:

$$\begin{aligned}
\left\|\exp(-H'/2) - \sum_{i=0}^{p}\frac{(H'/2)^i}{i!}\right\| = \left\|\sum_{i=p+1}^{\infty}\frac{(H'/2)^i}{i!}\right\| &\leq \sum_{j=p+1}^{\infty}\frac{\|H'\|^j}{j!} \leq \sum_{j=p+1}^{\infty}\Big(\frac{e\|H'\|}{j}\Big)^j \\
&\leq \Big(\frac{e\|H'\|}{p+1}\Big)^{p+1}\frac{1}{1-\Big(\frac{e\|H'\|}{(p+1)}\Big)} \leq \Big(\frac{1}{2}\Big)^p = \theta/\sqrt{n},
\end{aligned}$$

[8] For ease of notation, we write H for ηH.

[9] Here we assume that $\theta \leq 1$. Then, since $\lambda_{\min}(B) \geq 1$, we trivially have $\theta/\|u\| \leq 1/\sqrt{n} \leq \sqrt{n} \leq \|\sqrt{B}u\|$.

Lemma 33. *Given a Hermitian s-sparse matrix A, with $||A|| \leq 1$, a psd matrix $B = \exp(-H)$ with $H \preceq 0$, for a d-sparse H, and parameters $0 < \delta \leq 1/2$ and $0 < \theta \leq 1$. With probability $1 - \delta$, using $k = \mathcal{O}\Big(\log(\frac{1}{\delta})/\theta^2\Big)$ samples from the distribution $v^T Av$, one can find an estimate that is at most $\theta \mathrm{Tr}(B)$ away from $\mathrm{Tr}(AB)$. Here*

$$v = \sum_{i=0}^{p} \frac{(H/2)^i}{i!} u$$

where $p = \mathcal{O}(\max\{\|H\|, \log\Big(\frac{\sqrt{n}}{\theta}\Big)\})$, and $u = (u_j)$ where the $u_j \in \{\pm 1\}$ are i.i.d. uniformly distributed.

Lemma 34. *Given m Hermitian s-sparse $n \times n$ matrices $A_1 = I, A_2, \ldots, A_m$, with $\|A_j\| \leq 1$ for all j, a Hermitian d-sparse $n \times n$ matrix H with $\|H\| \leq K$, and parameters $0 < \delta \leq 1/2$ and $0 < \theta \leq 1$. With probability $1 - \delta$, we can compute θ-approximations $a_1, \ldots, a_m$ of $\mathrm{Tr}(A_1 B)/\mathrm{Tr}(B), \ldots, \mathrm{Tr}(A_m B)/\mathrm{Tr}(B)$ where $B = \exp(-H)$, using*

$$\mathcal{O}\left(\frac{\log(\frac{m}{\delta})}{\theta^2} \max\left\{K, \log\left(\frac{\sqrt{n}}{\theta}\right)\right\} dn + \frac{\log(\frac{m}{\delta})}{\theta^2} msn\right)$$

queries to the entries of $A_1, \ldots, A_m$, H and arithmetic operations.

Proof. As observed above, for every matrix A,

$$\frac{\mathrm{Tr}(A \exp(-H))}{\mathrm{Tr}(\exp(-H))} = \frac{\mathrm{Tr}(A \exp(-H + \gamma I))}{\mathrm{Tr}(\exp(-H + \gamma I))}.$$

Lemma 33 states that for $B' = \exp(-H + KI)$ and $A \in \{A_1, \ldots, A_m\}$ using $k = \mathcal{O}(\log(\frac{m}{\delta})/\theta^2)$ samples from the distribution $v^T Av$, one can find an estimate that is at most $\theta \mathrm{Tr}(B)$ away from $\mathrm{Tr}(AB)$ with probability $1 - \delta/m$. Here

$$v = \sum_{i=0}^{p} \frac{((H - KI)/2)^i}{i!} u$$

where $p = \mathcal{O}(\max\{K, \log\Big(\frac{\sqrt{n}}{\theta}\Big)\})$, and $u = (u_j)$ with the $u_j \in \{\pm 1\}$ i.i.d. uniformly distributed.

Observe that the k samples from $v^T Av$ are really obtained from k samples of vectors $u = (u_j)$ combined with some post-processing, namely obtaining $v = \sum_{i=0}^{p} \frac{((H-KI)/2)^i}{i!} u$ and two more sparse matrix vector products.

We can therefore obtain k samples from each of $v^T A_1 v, \ldots, v^T A_m v$ by *once* calculating k vectors $v = \sum_{i=0}^{p} \frac{((H-KI)/2)^i}{i!} u$, and then, for each of the m matrices A_j computing the k products $v^T A_j v$. The k vectors v can be constructed using

$$\mathcal{O}\left(\frac{\log(\frac{m}{\delta})}{\theta^2} \max\left\{K, \log\left(\frac{\sqrt{n}}{\theta}\right)\right\} dn\right)$$

queries to the entries of H and arithmetic operations. The mk matrix vector products can be computed using

$$\mathcal{O}\left(\frac{\log(\frac{m}{\delta})}{\theta^2} msn\right)$$

arithmetic operations and queries to the entries of $A_1, \ldots, A_m$ and H. This leads to total complexity

$$\mathcal{O}\left(\frac{\log(\frac{m}{\delta})}{\theta^2} \max\left\{K, \log\left(\frac{\sqrt{n}}{\theta}\right)\right\} dn + \frac{\log(\frac{m}{\delta})}{\theta^2} msn\right)$$

for computing k samples from each of $v^T A_1 v, \ldots, v^T A_m v$.

The results of Lemma 33 say that for each j, using those k samples of $v^T A_j v$ we can construct a $\theta \mathrm{Tr}(B')/4$-approximation a'_j of $\mathrm{Tr}(A_j B')$, with probability $1-\delta/(2m)$. Therefore, by a union bound, with probability $1-\delta/2$ we can construct $\theta \mathrm{Tr}(B')/4$-approximations $a'_1, \ldots, a'_m$ of $\mathrm{Tr}(A_1 B'), \ldots, \mathrm{Tr}(A_m B')$. Therefore, for each j, with probability at least $1-\delta$, by Lemma 7 we have that $a_j = a'_j/a'_1$ is a θ-approximation of $\mathrm{Tr}(A_j B')/\mathrm{Tr}(B')$, and hence it is a θ-approximation of $\mathrm{Tr}(A_j B)/\mathrm{Tr}(B)$. □

B Implementing smooth functions of Hamiltonians

In this appendix we show how to efficiently implement smooth functions of a given Hamiltonian. First we explain what we mean by a function of a Hamiltonian $H \in \mathbb{C}^{n \times n}$, i.e., a Hermitian matrix. Since Hermitian matrices are diagonalizable using a unitary matrix, we can write $H = U^\dagger \mathrm{diag}(\lambda) U$, where $\lambda \in \mathbb{R}^n$ is the vector of eigenvalues. Then for a function $f : \mathbb{R} \to \mathbb{C}$ we define $f(H) := U^\dagger \mathrm{diag}(f(\lambda)) U$ with a slight abuse of notation, where we apply f to the eigenvalues in λ one-by-one. Note that if we approximate f by $\tilde{f}$, then $\left\| \tilde{f}(H) - f(H) \right\| = \left\| \mathrm{diag}(\tilde{f}(\lambda)) - \mathrm{diag}(f(\lambda)) \right\|$. Suppose $D \subseteq \mathbb{R}$ is such that $\lambda \in D^n$, then we can upper bound this norm by the maximum of $|\tilde{f}(x) - f(x)|$ over $x \in D$. Finally we note that $D = [-\|H\|, \|H\|]$ is always a valid choice.

The main idea of the method presented below, is to implement a map $\tilde{f}(H)$, where $\tilde{f}$ is a good (finite) Fourier approximation of f for all $x \in [-\|H\|, \|H\|]$. The novelty in our approach is that we construct a Fourier approximation based on some polynomial approximation. In the special case, when f is analytic and $\|H\|$ is less than the radius of convergence of the Taylor series, we can obtain good polynomial approximation functions simply by truncating the Taylor series, with logarithmic dependence on the precision parameter. Finally we implement the Fourier series using Hamiltonian simulation and the Linear Combination of Unitaries (LCU) trick [CW12, BCC+15, BCK15].

This approach was already used in several earlier papers, particularly in [CKS17, CS17]. There the main technical difficulty was to obtain a good truncated Fourier series. This is a non-trivial task, since on top of the approximation error, one needs to optimize two other parameters of the Fourier approximation that determine the complexity of implementation, namely:

- the largest time parameter t that appears in some Fourier term e^{-itH}, and
- the total weight of the coefficients, by which we mean the 1-norm of the vector of coefficients.

Earlier works used clever integral approximations and involved calculus to construct a good Fourier approximation for a specific function f. We are not aware of a general result.

In contrast, our Theorem 40 and Corollary 42 avoids the usage of any integration. It obtains a low-weight Fourier approximation function using the Taylor series. The described method is completely general, and has the nice property that the maximal time parameter t depends logarithmically on the desired approximation precision. Since it uses the Taylor series, it is easy to apply to a wide range of smooth functions.

The circuit we describe for the implementation of the linear operator $f(H) : \mathbb{C}^n \to \mathbb{C}^n$ is going to depend on the specific function f, but not on H; the H-dependence is only coming from Hamiltonian simulation. Since the circuit for a specific f can be constructed in advance, we do not need to worry about the (polynomial) cost of constructing the circuit, making the analysis simpler. When we describe gate complexity, we count the number of two-qubit gates needed for a quantum circuit implementation, just as in Section 2.

Since this appendix presents stand-alone results, here we will deviate slightly from the notation used throughout the rest of the paper, to conform to the standard notation used in the literature (for example, ε, r, θ and a have a different meaning in this appendix). For simplicity we also assume, that the Hamiltonian H acts on $\mathbb{C}^n$, where n is a power of 2. Whenever we write log(formula) in some complexity statement we actually mean $\log_2(2 + \text{formula})$ in order to avoid incorrect near-0 or even negative expressions in complexity bounds that would appear for small values of the formula.

Hamiltonian simulation. We implement each term in a Fourier series using a Hamiltonian simulation algorithm, and combine the terms using the LCU Lemma. Specifically we use [BCK15], but in fact our techniques would work with any kind of Hamiltonian simulation algorithm.[10] The following definition describes what we mean by controlled Hamiltonian simulation.

Definition 35. *Let $M = 2^J$ for some $J \in \mathbb{N}$, $\gamma \in \mathbb{R}$ and $\epsilon \geq 0$. We say that the unitary*

$$W := \sum_{m=-M}^{M-1} |m\rangle\langle m| \otimes e^{im\gamma H}$$

implements controlled (M, γ)-simulation of the Hamiltonian H, where $|m\rangle$ denotes a (signed) bitstring $|b_J b_{J-1} \dots b_0\rangle$ such that $m = -b_J 2^J + \sum_{j=0}^{J-1} b_j 2^j$. The unitary $\tilde{W}$ implements controlled (M, γ, ε)-simulation of the Hamiltonian H, if

$$\left\| \tilde{W} - W \right\| \leq \varepsilon.$$

Note that in this definition we assume that both positive and negative powers of e^{iH} are simulated. This is necessary for our Fourier series, but sometimes we use only positive powers, e.g., for phase estimation; in that case we can simply ignore the negative powers.

The following lemma is inspired by the techniques of [CKS17]. It calculates the cost of such controlled Hamiltonian simulation in terms of queries to the input oracles (4)-(5) as described in Section 2.

Lemma 36. *Let $H \in \mathbb{C}^{n \times n}$ be a d-sparse Hamiltonian. Suppose we know an upper bound $K \in \mathbb{R}_+$ on the norm of H, i.e., $\|H\| \leq K$, and let $\tau := M\gamma K$. If $\varepsilon > 0$ and $\gamma = \Omega(1/(Kd))$, then a controlled (M, γ, ε)-simulation of H can be implemented using $\mathcal{O}(\tau d \log(\tau/\varepsilon)/\log\log(\tau/\varepsilon))$ queries and $\mathcal{O}\Big(\tau d \log(\tau/\varepsilon)/\log\log(\tau/\varepsilon)\Big[\log(n) + \log^{\frac{5}{2}}(\tau/\varepsilon)\Big]\Big)$ gates.*

Proof. We use the results of [BCK15, Lemma 9-10], which tell us that a d-sparse Hamiltonian H can be simulated for time t with ε precision in the operator norm using

$$\mathcal{O}\Big((t\|H\|_{\max} d + 1) \frac{\log(t\|H\|/\varepsilon)}{\log\log(t\|H\|/\varepsilon)}\Big) \tag{13}$$

queries and gate complexity

$$\mathcal{O}\Big((t\|H\|_{\max} d + 1) \frac{\log(t\|H\|/\varepsilon)}{\log\log(t\|H\|/\varepsilon)} \Big[\log(n) + \log^{\frac{5}{2}}(t\|H\|/\varepsilon)\Big]\Big). \tag{14}$$

[10] For example there is a more recent method for Hamiltonian simulation [LC19, LC17] that could possibly improve on some of the log factors we get from [BCK15], but one could even consider completely different input models allowing different simulation methods.

Now we use a standard trick to remove log factors from the implementation cost, and write the given unitary W as the product of some increasingly precisely implemented controlled Hamiltonian simulation unitaries. For $b \in \{0,1\}$ let us introduce the projector $|b\rangle\langle b|_j := I_{2^j} \otimes |b\rangle\langle b| \otimes I_{2^{J-j}}$, where $J = \log(M)$. Observe that

$$W = \Big(|1\rangle\langle 1|_J \otimes e^{-i2^J\gamma H} + |0\rangle\langle 0|_J \otimes I\Big) \prod_{j=0}^{J-1} \Big(|1\rangle\langle 1|_j \otimes e^{i2^j\gamma H} + |0\rangle\langle 0|_j \otimes I\Big). \tag{15}$$

The j-th operator $e^{\pm i2^j\gamma H}$ in the product (15) can be implemented with $2^{j-J-1}\epsilon$ precision using $\mathcal{O}\Big(2^j\gamma K d \log\Big(\frac{2^j\gamma K}{\varepsilon 2^{j-J-1}}\Big)/\log\log\Big(\frac{2^j\gamma K}{\varepsilon 2^{j-J-1}}\Big)\Big) = \mathcal{O}(2^j\gamma K d \log(\tau/\epsilon)/\log\log(\tau/\epsilon))$ queries by (13) and using $\mathcal{O}(2^j d \log(\tau/\epsilon)/\log\log(\tau/\epsilon)[\log(n) + \log^{\frac{5}{2}}(\tau/\epsilon)])$ gates by (14). Let us denote by $\tilde{W}$ the concatenation of all these controlled Hamiltonian simulation unitaries. Adding up the costs we see that our implementation of $\tilde{W}$ uses $\mathcal{O}(\tau d \log(\tau/\varepsilon)/\log\log(\tau/\varepsilon))$ queries and has gate complexity $\mathcal{O}\Big(\tau d \log(\tau/\varepsilon)/\log\log(\tau/\varepsilon)\Big[\log(n) + \log^{\frac{5}{2}}(\tau/\varepsilon)\Big]\Big)$. Using the triangle inequality repeatedly, it is easy to see that $\left\|W - \tilde{W}\right\| \leq \sum_{j=0}^{J} 2^{j-J-1}\varepsilon \leq \varepsilon$. □

B.1 Implementation of smooth functions of Hamiltonians: general results

The first lemma we prove provides the basis for our approach. It shows how to turn a polynomial approximation of a function f on the interval $[-1,1]$ into a nice Fourier series in an efficient way, while not increasing the weight of coefficients. This is useful, because we can implement a function given by a Fourier series using the LCU Lemma, but only after scaling it down with the weight of the coefficients.

Lemma 37. *Let $\delta, \varepsilon \in (0,1)$ and $f : \mathbb{R} \to \mathbb{C}$ s.t. $\left|f(x) - \sum_{k=0}^{K} a_k x^k\right| \leq \varepsilon/4$ for all $x \in [-1+\delta, 1-\delta]$. Then $\exists\, c \in \mathbb{C}^{2M+1}$ such that*

$$\left| f(x) - \sum_{m=-M}^{M} c_m e^{\frac{i\pi m}{2} x} \right| \leq \varepsilon$$

for all $x \in [-1+\delta, 1-\delta]$, where $M = \max\Big(2\Big\lceil \ln\Big(\frac{4\|a\|_1}{\varepsilon}\Big)\frac{1}{\delta}\Big\rceil, 0\Big)$ and $\|c\|_1 \leq \|a\|_1$. Moreover c can be efficiently calculated on a classical computer in time poly$(K, M, \log(1/\varepsilon))$.

Proof. Let us introduce the notation $\|f\|_\infty = \sup\{|f(x)| : x \in [-1+\delta, 1-\delta]\}$. First we consider the case when $\|a\|_1 < \varepsilon/2$. Then $\|f\|_\infty \leq \left\|f(x) - \sum_{k=0}^{K} a_k x^k\right\|_\infty + \left\|\sum_{k=0}^{K} a_k x^k\right\|_\infty < \varepsilon/4 + \varepsilon/2 < \varepsilon$. So in this case the statement holds with $M = 0$ and $c = 0$, i.e., even with an empty sum.

From now on we assume $\|a\|_1 \geq \varepsilon/2$. We are going to build up our approximation gradually. Our first approximate function $\tilde{f}_1(x) := \sum_{k=0}^{K} a_k x^k$ satisfies $\left\|f - \tilde{f}_1\right\|_\infty \leq \varepsilon/4$ by assumption. In order to construct a Fourier series, we will work towards a linear combination of sines. To that end, note that $\forall x \in [-1,1]$: $\tilde{f}_1(x) = \sum_{k=0}^{K} a_k \Big(\frac{\arcsin(\sin(x\pi/2))}{\pi/2}\Big)^k$. Let $b^{(k)}$ denote the series of coefficients such that $\Big(\frac{\arcsin(y)}{\pi/2}\Big)^k = \sum_{\ell=0}^{\infty} b_\ell^{(k)} y^\ell$ for all $y \in [-1,1]$. For $k = 1$ the coefficients are just $\frac{2}{\pi}$ times the coefficients of the Taylor series of arcsin so we know that $b_{2\ell}^{(1)} = 0$ while $b_{2\ell+1}^{(1)} = \binom{2\ell}{\ell}\frac{2^{-2\ell}}{2\ell+1}\frac{2}{\pi}$. Since $\Big(\frac{\arcsin(y)}{\pi/2}\Big)^{k+1} = \Big(\frac{\arcsin(y)}{\pi/2}\Big)^k \Big(\sum_{\ell=0}^{\infty} b_\ell^{(1)} y^\ell\Big)$, we obtain the formula $b_\ell^{(k+1)} = \sum_{\ell'=0}^{\ell} b_{\ell'}^{(k)} b_{\ell-\ell'}^{(1)}$, so one can recursively calculate each $b^{(k)}$. As $b^{(1)} \geq 0$ one can

use the above identity inductively to show that $b^{(k)} \geq 0$. Therefore $\left\|b^{(k)}\right\|_1 = \sum_{\ell=0}^{\infty} b_\ell^{(k)} 1^\ell = \left(\frac{\arcsin(1)}{\pi/2}\right)^k = 1$. Using the above definitions and observations we can rewrite

$$\forall x \in [-1,1] : \tilde{f}_1(x) = \sum_{k=0}^{K} a_k \sum_{\ell=0}^{\infty} b_\ell^{(k)} \sin^\ell(x\pi/2).$$

To obtain the second approximation function, we want to truncate the summation over ℓ at $L = \ln\left(\frac{4\|a\|_1}{\varepsilon}\right)\frac{1}{\delta^2}$ in the above formula. We first estimate the tail of the sum. We are going to use that for all $\delta \in [0,1]$: $\sin((1-\delta)\pi/2) \leq 1-\delta^2$. For all $k \in \mathbb{N}$ and $x \in [-1+\delta, 1-\delta]$ we have:

$$\begin{aligned}
\left|\sum_{\ell=\lceil L\rceil}^{\infty} b_\ell^{(k)} \sin^\ell(x\pi/2)\right| &\leq \sum_{\ell=\lceil L\rceil}^{\infty} b_\ell^{(k)} \left|\sin^\ell(x\pi/2)\right| \\
&\leq \sum_{\ell=\lceil L\rceil}^{\infty} b_\ell^{(k)} \left|1-\delta^2\right|^\ell \\
&\leq \left(1-\delta^2\right)^L \sum_{\ell=\lceil L\rceil}^{\infty} b_\ell^{(k)} \\
&\leq \left(1-\delta^2\right)^L \\
&\leq e^{-\delta^2 L} \\
&= \frac{\varepsilon}{4\|a\|_1}.
\end{aligned}$$

Thus we have $\left\|\tilde{f}_1 - \tilde{f}_2\right\|_\infty \leq \varepsilon/4$ for

$$\tilde{f}_2(x) := \sum_{k=0}^{K} a_k \sum_{\ell=0}^{\lfloor L\rfloor} b_\ell^{(k)} \sin^\ell(x\pi/2).$$

To obtain our third approximation function, we will approximate $\sin^\ell(x\pi/2)$. First observe that

$$\sin^\ell(z) = \left(\frac{e^{-iz} - e^{iz}}{-2i}\right)^\ell = \left(\frac{i}{2}\right)^\ell \sum_{m=0}^{\ell} (-1)^m \binom{\ell}{m} e^{iz(2m-\ell)} \tag{16}$$

which, as we will show (for M' much larger than $\sqrt{\ell}$) is very well approximated by

$$\left(\frac{i}{2}\right)^\ell \sum_{m=\lceil \ell/2\rceil - M'}^{\lfloor \ell/2\rfloor + M'} (-1)^m \binom{\ell}{m} e^{iz(2m-\ell)}.$$

Truncating the summation in (16) based on this approximation reduces the maximal time evolution parameter (i.e., the maximal value of the parameter t in the $\exp(izt)$ terms) quadratically. To make this approximation precise, we use Chernoff's inequality [AS08, A.1.7] for the binomial distribution, or more precisely its corollary for sums of binomial coefficients, stating

$$\sum_{m=\lceil \ell/2 + M'\rceil}^{\ell} 2^{-\ell} \binom{\ell}{m} \leq e^{-\frac{2(M')^2}{\ell}}.$$

Let $M' = \left\lceil \ln\left(\frac{4\|a\|_1}{\varepsilon}\right)\frac{1}{\delta}\right\rceil$ and suppose $\ell \le L$, then this bound implies that

$$\sum_{m=0}^{\lfloor \ell/2\rfloor - M'} 2^{-\ell}\binom{\ell}{m} = \sum_{m=\lceil \ell/2\rceil + M'}^{\ell} 2^{-\ell}\binom{\ell}{m} \le e^{-\frac{2(M')^2}{\ell}} \le e^{-\frac{2(M')^2}{L}} \le \left(\frac{\varepsilon}{4\|a\|_1}\right)^2 \le \frac{\varepsilon}{4\|a\|_1}, \quad (17)$$

where for the last inequality we use the assumption $\varepsilon \le 2\|a\|_1$. By combining (16) and (17) we get that for all $\ell \le L$

$$\left\| \sin^\ell(z) - \left(\frac{i}{2}\right)^\ell \sum_{m=\lceil \ell/2\rceil - M'}^{\lfloor \ell/2\rfloor + M'} (-1)^m \binom{\ell}{m} e^{iz(2m-\ell)} \right\|_\infty \le \frac{\varepsilon}{2\|a\|_1}.$$

Substituting $z = x\pi/2$ into this bound we can see that $\left\|\tilde{f}_2 - \tilde{f}_3\right\|_\infty \le \varepsilon/2$, for

$$\tilde{f}_3(x) := \sum_{k=0}^{K} a_k \sum_{\ell=0}^{\lfloor L\rfloor} b_\ell^{(k)} \left(\frac{i}{2}\right)^\ell \sum_{m=\lceil \ell/2\rceil - M'}^{\lfloor \ell/2\rfloor + M'} (-1)^m \binom{\ell}{m} e^{\frac{i\pi x}{2}(2m-\ell)}, \quad (18)$$

using $\sum_{k=0}^{K} |a_k| \sum_{\ell=0}^{\lfloor L\rfloor} \left|b_\ell^{(k)}\right| \le \sum_{k=0}^{K} |a_k| = \|a\|_1$. Therefore we can conclude that $\tilde{f}_3$ is an ε-approximation to f:

$$\left\|f - \tilde{f}_3\right\|_\infty \le \left\|f - \tilde{f}_1\right\|_\infty + \left\|\tilde{f}_1 - \tilde{f}_2\right\|_\infty + \left\|\tilde{f}_2 - \tilde{f}_3\right\|_\infty \le \varepsilon.$$

Observe that in (18) the largest value of $|m-\ell|$ in the exponent is upper bounded by $2M' = M$. So by rearranging the terms in $\tilde{f}_3$ we can write $\tilde{f}_3(x) = \sum_{m=-M}^{M} c_m e^{\frac{i\pi m}{2}x}$. Now let us fix a value k in the first summation of (18). Observe that after taking the absolute value of each term, the last two summations still yield a value ≤ 1, since $\left\|b^{(k)}\right\|_1 = 1$ and $\sum_{m=0}^{\ell}\binom{\ell}{m} = 2^\ell$. It follows that $\|c\|_1 \le \|a\|_1$. From the construction of the proof, it is easy to see that (an ε-approximation of) c can be calculated in time $\mathrm{poly}(K, M, \log(1/\varepsilon))$. □

Now we present the Linear Combination of Unitaries (LCU) Lemma [CW12, BCC+15, BCK15], which we will use for combining the Fourier terms in our quantum circuit. Since we intend to use LCU for implementing non-unitary operations, we describe a version without the final amplitude amplification step. We provide a short proof for completeness.

Lemma 38 (LCU Lemma [CW12, BCC+15, BCK15])**.** *Let $U_1, U_2, \ldots, U_m$ be unitaries on a Hilbert space $\mathcal{H}$, and $L = \sum_{i=1}^m a_i U_i$, where $a \in \mathbb{R}_+^m \setminus \{0\}$. Let $V = \sum_{i=1}^m |i\rangle\langle i| \otimes U_i$ and $A \in \mathbb{C}^{m\times m}$ be a unitary such that $A|0\rangle = \sum_{i=1}^m \sqrt{\frac{a_i}{\|a\|_1}}|i\rangle$. Then $\frac{L}{\|a\|_1} = (\langle 0| \otimes I)\left(A^\dagger \otimes I\right)V(A \otimes I)(|0\rangle \otimes I)$, i.e., for every $|\psi\rangle \in \mathcal{H}$ we have $\left(A^\dagger \otimes I\right)V(A \otimes I)|0\rangle|\psi\rangle = |0\rangle\frac{L}{\|a\|_1}|\psi\rangle + |\Phi^\perp\rangle$, where the vector $|\Phi^\perp\rangle$ satisfies $(|0\rangle\langle 0| \otimes I)|\Phi^\perp\rangle = 0$.*

Proof.

$$\begin{aligned}
(\langle 0| \otimes I)(A^\dagger \otimes I)V(A \otimes I)|0\rangle|\psi\rangle &= \left(\left(\sum_{i=1}^m \sqrt{\frac{a_i}{\|a\|_1}}\langle i|\right) \otimes I\right) V \sum_{i=1}^m \sqrt{\frac{a_i}{\|a\|_1}}|i\rangle|\psi\rangle \\
&= \left(\left(\sum_{i=1}^m \sqrt{\frac{a_i}{\|a\|_1}}\langle i|\right) \otimes I\right) \sum_{i=1}^m \sqrt{\frac{a_i}{\|a\|_1}}|i\rangle U_i|\psi\rangle \\
&= \sum_{i=1}^m \frac{a_i}{\|a\|_1} U_i|\psi\rangle \\
&= \frac{L}{\|a\|_1}|\psi\rangle
\end{aligned}$$

□

The next result summarizes how to efficiently implement a Fourier series of a Hamiltonian.

Lemma 39. *Suppose $f(x) = \sum_{m=-M}^{M-1} c_m e^{im\gamma x}$, for a given $c \in \mathbb{C}^{2M} \setminus \{0\}$. We can construct a unitary $\tilde{U}$ which implements the operator $\frac{f(H)}{\|c\|_1} = \sum_{m=-M}^{M-1} \frac{c_m}{\|c\|_1} e^{im\gamma H}$ with ε precision, i.e., such that*

$$\left\| (\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - \frac{f(H)}{\|c\|_1} \right\| \leq \varepsilon,$$

using $\mathcal{O}(M(\log(M)+1))$ two-qubit gates and a single use of a circuit implementing controlled (M, γ, ε)-simulation of H.

Proof. This is a direct corollary of Lemma 38. To work out the details, note that we can always extend c with some 0 values, so we can assume without loss of generality that M is a power of 2. This is useful, because then we can represent each $m \in [-M, M-1]$ as a $(J+1)$-bit signed integer for $J = \log(M)$.

The implementation of the operator A in Lemma 38 does not need any queries and it can be constructed exactly using $\mathcal{O}(M(\log(M)+1))$ two-qubit gates, e.g., by the techniques of [GR02]. We sketch the basic idea which is based on induction. For $J = 1$ the operator A is just a two-qubit unitary. Suppose we proved the claim for bitstrings of length J and want to prove the claim for length $J+1$. Let $a \in \mathbb{R}_+^{2^{J+1}}$ be such that $\|a\|_1 = 1$ and define $\tilde{a} \in \mathbb{R}_+^{2^J}$ such that $\tilde{a}_b = a_{b,0} + a_{b,1}$ for all bitstrings $b \in \{0,1\}^J$. Then we have a circuit $\tilde{A}$ that uses $\mathcal{O}(2^J(J+1))$ gates and satisfies $\sqrt{\tilde{a}_b} = \langle b|\tilde{A}|0\ldots0\rangle$ for all $b \in \{0,1\}^J$. We can add an extra $|0\rangle$-qubit and implement a controlled rotation gate R_b on it for each $b \in \{0,1\}^J$. Let R_b have rotation angle $\arccos\left(\sqrt{a_{b,0}/\tilde{a}_b}\right)$ and be controlled by b. It is easy to see that the new unitary A satisfies $\sqrt{a_{b'}} = \langle b'|A|0\ldots0\rangle$ for each $b' \in \{0,1\}^J$. Each R_b can be implemented using $\mathcal{O}(J)$ two-qubit gates and ancilla qubits, justifying the gate complexity and concluding the induction.

What remains is to implement the operator $V = \sum_{m=-M}^{M-1} |m\rangle\langle m| \otimes \frac{c_m}{|c_m|} e^{im\gamma H}$ from Lemma 38. We implement $V = PW$ in two steps, where $P = \sum_{m=-M}^{M-1} |m\rangle\langle m| \otimes \frac{c_m}{|c_m|} I$. This P can be implemented exactly using $\mathcal{O}(M(\log(M)+1))$ gates simply by building a controlled gate that adds the right phase for each individual bitstring. Since the bitstring on which we want to do a controlled operation has length $\log(M)+1$, each controlled operation can be constructed using $\mathcal{O}(\log(M)+1)$ gates and ancilla qubits resulting in the claimed gate complexity. We use a circuit implementing controlled (M, γ, ε)-simulation of H, denoted by $\tilde{W}$, which is an ε-approximation of W by definition.

Finally $\tilde{U} := (A^\dagger \otimes I)P\tilde{W}(A \otimes I)$. This yields an ε-precise implementation, since

$$\begin{aligned}
\left\| (\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - \frac{f(H)}{\|c\|_1} \right\| &= \left\| (\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - (\langle 0| \otimes I)(A^\dagger \otimes I)PW(A \otimes I)(|0\rangle \otimes I) \right\| \\
&\leq \left\| \tilde{U} - (A^\dagger \otimes I)PW(A \otimes I) \right\| \\
&= \left\| (A^\dagger \otimes I)P\tilde{W}(A \otimes I) - (A^\dagger \otimes I)PW(A \otimes I) \right\| \\
&= \left\| \tilde{W} - W \right\| \leq \varepsilon.
\end{aligned}$$

□

Now we can state the main result of this appendix, which tells us how to efficiently turn a function (provided with its Taylor series) of a Hamiltonian H, into a quantum circuit by using controlled Hamiltonian simulation.

In the following theorem we assume that the eigenvalues of H lie in a radius-r ball around x_0. The main idea is that if even $r + \delta$ is less than the radius of convergence of the Taylor

series, then we can obtain an ε-approximation of f by truncating the series at logarithmically high powers. B will be an upper bound on the absolute value of the function within the $r+\delta$ ball around x_0, in particular $\|f(H)/B\| \leq 1$. Therefore we can implement $f(H)/B$ as a block of some larger unitary. It turns out that apart from the norm and sparsity of H and precision parameters, the complexity depends on the ratio of δ and r.

Theorem 40 (Implementing a smooth function of a Hamiltonian)**.** *Let $x_0 \in \mathbb{R}$ and $r > 0$ be such that $f(x_0+x) = \sum_{\ell=0}^{\infty} a_\ell x^\ell$ for all $x \in [-r,r]$. Suppose $B > 0$ and $\delta \in (0,r]$ are such that $\sum_{\ell=0}^{\infty}(r+\delta)^\ell |a_\ell| \leq B$. If $\|H - x_0 I\| \leq r$ and $\varepsilon \in \left(0, \frac{1}{2}\right]$, then we can implement a unitary $\tilde{U}$ such that $\left\|(\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - \frac{f(H)}{B}\right\| \leq \varepsilon$, using $\mathcal{O}(r/\delta \log(r/(\delta\varepsilon)) \log(1/\varepsilon))$ gates and a single use of a circuit for controlled $(\mathcal{O}(r\log(1/\varepsilon)/\delta), \mathcal{O}(1/r), \varepsilon/2)$-simulation of H.*

Suppose we are given K such that $\|H\| \leq K$ and $r = \mathcal{O}(K)$. If, furthermore, H is d-sparse and is accessed via oracles (4)-(5), then the whole circuit can be implemented using

$$\mathcal{O}\left(\frac{Kd}{\delta}\log\left(\frac{K}{\delta\varepsilon}\right)\log\left(\frac{1}{\varepsilon}\right)\right) \text{ queries and } \mathcal{O}\left(\frac{Kd}{\delta}\log\left(\frac{K}{\delta\varepsilon}\right)\log\left(\frac{1}{\varepsilon}\right)\left[\log(n)+\log^{\frac{5}{2}}\left(\frac{K}{\delta\varepsilon}\right)\right]\right) \text{ gates.}$$

Proof. The basic idea is to combine Lemma 37 and Lemma 39 and apply them to a transformed version of the function. First we define $\delta' := \delta/(r+\delta)$, which is at most $1/2$ by assumption. Then, for all $\ell \in \mathbb{N}$ let $b_\ell := a_\ell (r+\delta)^\ell$ and define the function $g : [-1+\delta', 1-\delta'] \to \mathbb{R}$ by $g(y) := \sum_{\ell=0}^{\infty} b_\ell y^\ell$ so that

$$f(x_0+x) = g(x/(r+\delta)) \qquad \text{for all } x \in [-r,r]. \tag{19}$$

Now we set $L := \left\lceil \frac{1}{\delta'} \log\left(\frac{8}{\varepsilon}\right)\right\rceil$. Then for all $y \in [-1+\delta', 1-\delta']$

$$\begin{aligned}
\left|g(y) - \sum_{\ell=0}^{L-1} b_\ell y^\ell\right| &= \left|\sum_{\ell=L}^{\infty} b_\ell y^\ell\right| \\
&\leq \sum_{\ell=L}^{\infty}\left|b_\ell (1-\delta')^\ell\right| \\
&\leq (1-\delta')^L \sum_{\ell=L}^{\infty} |b_\ell| \\
&\leq (1-\delta')^L B \\
&\leq e^{-\delta' L} B \\
&\leq \frac{\varepsilon B}{8}.
\end{aligned}$$

We would now like to obtain a Fourier-approximation of g for all $y \in [-1+\delta', 1-\delta']$, with precision $\varepsilon' = \frac{\varepsilon B}{2}$. Let $b' := (b_0, b_1, \ldots, b_{L-1})$ and observe that $\|b'\|_1 \leq \|b\|_1 \leq B$. We apply Lemma 37 to the function g, using the polynomial approximation corresponding to the truncation to the first L terms, i.e., using the coefficients in b'. Then we obtain a Fourier ε'-approximation $\tilde{g}(y) := \sum_{m=-M}^{M} \tilde{c}_m e^{\frac{i\pi m}{2} y}$ of g, with

$$M = \mathcal{O}\left(\frac{1}{\delta'}\log\left(\frac{\|b'\|_1}{\varepsilon'}\right)\right) = \mathcal{O}\left(\frac{r}{\delta}\log\left(\frac{1}{\varepsilon}\right)\right)$$

such that the vector of coefficients $\tilde{c} \in \mathbb{C}^{2M+1}$ satisfies $\|\tilde{c}\|_1 \leq \|b'\|_1 \leq \|b\|_1 \leq B$. Let

$$\tilde{f}(x_0+x) := \tilde{g}\left(\frac{x}{r+\delta}\right) = \sum_{m=-M}^{M} \tilde{c}_m e^{\frac{i\pi m}{2(r+\delta)} x};$$

by (19) we see that $\tilde{f}$ is an ε'-precise Fourier approximation of f on the interval $[x_0 - r, x_0 + r]$. To transform this Fourier series to its final form, we note that $\tilde{f}(z) = \sum_{m=-M}^{M} \tilde{c}_m e^{\frac{i\pi m}{2(r+\delta)}(z-x_0)}$, so by defining $c_m := \tilde{c}_m e^{-\frac{i\pi m}{2(r+\delta)}x_0}$ we get a Fourier series in z, while preserving $\|c\|_1 = \|\tilde{c}\|_1 \leq B$.

In the trivial case, when $c = 0$, we choose a unitary $\tilde{U}$, such that it maps the $|0\rangle$ ancilla state to $|1\rangle$, then clearly $(\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) = 0 = \tilde{f}(H)$. Clearly such a $\tilde{U}$ can be implemented using $\mathcal{O}(1)$ gates and 0 queries. Otherwise we can apply Lemma 39 to this modified Fourier series to construct a unitary circuit $\tilde{V}$ implementing an $\frac{\varepsilon}{2}$-approximation of $\tilde{f}(H)/\|c\|_1$. We can further scale down the amplitude of the $|0\rangle$-part of the output by a factor of $\|c\|_1/B \leq 1$, to obtain an approximation of $\tilde{f}(H)/B$ as follows. We simply add an additional ancilla qubit initialized to $|0\rangle$ on which we act with the one-qubit unitary

$$Rot := \begin{pmatrix} \frac{\|c\|_1}{B} & \sqrt{1 - \frac{\|c\|_1^2}{B^2}} \\ -\sqrt{1 - \frac{\|c\|_1^2}{B^2}} & \frac{\|c\|_1}{B} \end{pmatrix}.$$

Finally we define $\tilde{U} := Rot \otimes \tilde{V}$, and define $|0\rangle|0\rangle$ as the new success indicator, where the first qubit is the new ancilla. We show that $\tilde{U}$ implements $f(H)/B$ with ε precision: (if $c = 0$, let us use the definition $\tilde{f}(H)/\|c\|_1 := 0$)

$$\begin{aligned}
\left\|(\langle 0|\langle 0| \otimes I)\tilde{U}(|0\rangle|0\rangle \otimes I) - \frac{f(H)}{B}\right\| &\leq \left\|(\langle 0|\langle 0| \otimes I)\tilde{U}(|0\rangle|0\rangle \otimes I) - \frac{\tilde{f}(H)}{B}\right\| + \left\|\frac{\tilde{f}(H)}{B} - \frac{f(H)}{B}\right\| \\
&= \frac{\|c\|_1}{B}\left\|(\langle 0| \otimes I)\tilde{V}(|0\rangle \otimes I) - \frac{\tilde{f}(H)}{\|c\|_1}\right\| + \left\|\frac{\tilde{f}(H) - f(H)}{B}\right\| \\
&\leq \frac{\|c\|_1}{B}\frac{\varepsilon}{2} + \frac{\varepsilon'}{B} \\
&\leq \varepsilon.
\end{aligned}$$

Lemma 39 uses $\mathcal{O}(M \log(M+1)) = \mathcal{O}(r/\delta \log(1/\varepsilon) \log(r/(\delta\varepsilon)))$ gates and a single use of a controlled $(M, \gamma = \pi/(2r + 2\delta), \varepsilon/2)$-simulation of H. If $\|H\| = \mathcal{O}(K)$, we can use Lemma 36 to conclude $\mathcal{O}\Big(M\gamma K d \log\Big(\frac{1}{\varepsilon}\Big) \log\Big(\frac{M\gamma K}{\varepsilon} \log\Big(\frac{1}{\varepsilon}\Big)\Big)\Big) = \mathcal{O}\Big(\frac{Kd}{\delta} \log\Big(\frac{K}{\delta\varepsilon}\Big) \log\Big(\frac{1}{\varepsilon}\Big)\Big)$ query and

$$\begin{aligned}
&\mathcal{O}\Big(M\gamma K d \log\Big(\frac{1}{\varepsilon}\Big) \log\Big(\frac{M\gamma K}{\varepsilon} \log\Big(\frac{1}{\varepsilon}\Big)\Big)\Big[\log(n) + \log^{\frac{5}{2}}\Big(\frac{M\gamma K}{\varepsilon} \log\Big(\frac{1}{\varepsilon}\Big)\Big)\Big]\Big) \\
&\qquad = \mathcal{O}\Big(\frac{Kd}{\delta} \log\Big(\frac{K}{\delta\varepsilon}\Big) \log\Big(\frac{1}{\varepsilon}\Big)\Big[\log(n) + \log^{\frac{5}{2}}\Big(\frac{K}{\delta\varepsilon}\Big)\Big]\Big)
\end{aligned}$$

gate complexity. Finally note that the polynomial cost of calculating c that is required by Lemma 37 does not affect the query complexity or the circuit size, it only affects the description of the circuit. □

Remark 41. *Note that in the above theorem we can relax the criterion $\|H - x_0 I\| \leq r$. Suppose we have an orthogonal projector Π, which projects to eigenvectors with eigenvalues in $[x_0 - r, x_0 + r]$, i.e., $[H, \Pi] = 0$ and $\|\Pi(H - x_0 I)\Pi\| \leq r$. Then the circuit $\tilde{U}$ constructed in Theorem 40 satisfies*

$$\left\|\Pi\Big((\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - \frac{f(H)}{B}\Big)\Pi\right\| \leq \varepsilon.$$

The following corollary shows how to implement functions piecewise in small "patches" using Remark 41. The main idea is to first estimate the eigenvalues of H up to θ precision, and then implement the function using the Taylor series centered around a point close to the eigenvalue.

This approach has multiple advantages. First, the function may not have a Taylor series that is convergent over the whole domain of possible eigenvalues of H. Even if there is such a series, it can have very poor convergence properties, making B large and therefore requiring a lot of amplitude amplification. Nevertheless, for small enough neighborhoods the Taylor series always converges quickly, overcoming this difficulty.

Corollary 42. *Suppose $(x_\ell) \in \mathbb{R}^L$ and $r, \theta \in \mathbb{R}_+$ are such that the spectrum of H lies in the domain $\bigcup_{\ell=1}^{L}[x_\ell - (r-2\theta), x_\ell + (r+2\theta)]$.[11] Suppose there exist coefficients $a_k^{(\ell)} \in \mathbb{R}$ such that for all $\ell \in [L]$ and $x \in [-r, r]$ we have $f(x_\ell + x) = \sum_{k=0}^{\infty} a_k^{(\ell)} x^k$, and $\sum_{k=0}^{\infty}(r+\delta)^k |a_k^{(\ell)}| \leq B$ for some fixed $\delta \in [0, r]$ and $B > 0$. If $\|H\| \leq K$ and $\varepsilon \in \left(0, \frac{1}{2}\right]$, then we can implement a unitary $\tilde{U}$ such that*

$$\left\| (\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - \frac{f(H)}{B} \right\| \leq \varepsilon,$$

using $\mathcal{O}(Lr/\delta \log(r/(\delta\varepsilon)) \log(1/\varepsilon) + \log(K/\theta) \log\log(K/(\theta\varepsilon)))$ gates, and with $\mathcal{O}(\log(1/\varepsilon))$ uses of an $(\mathcal{O}(1/\theta), \pi/K, \Omega(\varepsilon^2/\log(1/\varepsilon)))$-simulation of H and a single use of a circuit for controlled $(\mathcal{O}(r\log(1/\varepsilon)/\delta), \mathcal{O}(1/r), \varepsilon/2)$-simulation of H. If $r = \mathcal{O}(K)$, $\theta \leq r/4$, $\theta = \Omega(\delta)$, $\|H\| \leq K$, H is d-sparse and is accessed via oracles (4)-(5), then the circuit can be implemented using

$$\mathcal{O}\left(\frac{Kd}{\delta} \log\left(\frac{K}{\delta\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right)\right) \text{ queries and } \mathcal{O}\left(\frac{Kd}{\delta} \log\left(\frac{K}{\delta\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right)\left[\log(n) + \log^{\frac{5}{2}}\left(\frac{K}{\delta\varepsilon}\right)\right]\right) \text{ gates.}$$

Sketch of the proof. We start by performing phase estimation on e^{iH} with $\approx \theta$ resolution in phase. We boost the success probability by taking the median outcome of $\mathcal{O}(\log(1/\varepsilon))$ parallel repetitions, so that we get a worse-than-θ estimation with probability at most $\mathcal{O}(\varepsilon^2)$. This way the boosted phase estimation circuit is $\mathcal{O}(\varepsilon)$-close in operator norm to an "idealized" phase estimation unitary that never makes approximation error greater than θ (for more details on this type of argument, see the proof of Lemma 50). Phase estimation uses controlled $(\mathcal{O}(K/\theta), \pi/K, \Omega(\varepsilon^2/\log(1/\varepsilon)))$-simulation of H and a Fourier transform on $\mathcal{O}(\log(K/\theta))$-bit numbers which can be implemented using $\mathcal{O}(\log(K/\theta)\log\log(K/(\theta\varepsilon)))$ gates. The probability-boosting uses $\mathcal{O}(\log(1/\varepsilon))$ repetitions. Controlled on the phase estimate $\tilde{\lambda}$ that we obtained, we implement a $1/B$-scaled version of the corresponding function "patch" $f(x)|_{[x_\ell - r, x_\ell + r]}$ centered around $\arg\min |x_\ell - \tilde{\lambda}|$ using Theorem 40 and Remark 41. The additional gate complexities of the "patches" add up to $\mathcal{O}(Lr/\delta \log(r/(\delta\varepsilon)) \log(1/\varepsilon))$, but since each "patch" uses the same controlled $(\mathcal{O}(r\log(1/\varepsilon)/\delta), \mathcal{O}(1/r), \varepsilon/2)$-simulation of H, we only need to implement that once. Finally we uncompute phase estimation.[12] For the final complexity, note that we can assume without loss of generality that $L(r - 2\theta) = \mathcal{O}(K)$, since otherwise we can just remove some redundant intervals from the domain. Hence $Lr = \mathcal{O}(K)$ and Lemma 36 implies the stated complexities. □

This corollary is essentially as general and efficient as we can hope for. Let D denote the domain of possible eigenvalues of H. If we want to implement a reasonably smooth function f, then it probably satisfies the following: there is some $r = \Omega(1)$, such that for each $x \in D$,

[11] This way, even if we make θ error during the estimation of an eigenvalue λ_i, the closest x_ℓ will still contain λ_i in its radius-r neighborhood.

[12] Note that phase estimation on some eigenvector of e^{iH} can produce a superposition of different estimates of the phase. If some intervals $[x_\ell - r, x_\ell + r]$ overlap for ℓ and ℓ', those estimates could lead to different implementations of $f(H)$ (one based on the coefficients $a_k^{(\ell)}$ and one based on $a_k^{(\ell')}$). However, this causes no difficulty; since we used the same normalization $1/B$ for all implementations, both implementations lead to essentially the same state after postselecting on the $|0\rangle$ ancilla state.

the Taylor series in the radius-r neighborhood of x converges quickly, more precisely the Taylor coefficients $a_k^{(x)}$ for the x-centered series satisfy $\sum_{k=0}^{\infty} |a_k^{(x)}| r^k = \mathcal{O}(\|f\|_\infty)$, where we define $\|f\|_\infty := \sup_{x \in D} |f(x)|$. If this is the case, covering D with radius-$\mathcal{O}(r)$ intervals, choosing $\theta = \Theta(r)$ and $\delta = \Theta(r)$, Corollary 42 provides an $\tilde{\mathcal{O}}(\|H\| d)$ query and gate complexity implementation of $f(H)/B$, where $B = \mathcal{O}(\|f\|_\infty)$. The value of B is optimal up to constant factors, since $f(H)/B$ must have norm at most 1. Also the $\|H\| d$ factor in the complexity is very reasonable, and we achieve the logarithmic error dependence which is the primary motivation of the related techniques. An application along the lines of this discussion can be found in Lemma 46.

Also note that in the above corollary we added up the gate complexities of the different "patches." Since these gates prepare the Fourier coefficients of the function corresponding to the different Taylor series at different points, one could use this structure to implement all coefficients with a single circuit. This can potentially result in much smaller circuit sizes, which could be beneficial when the input model allows more efficient Hamiltonian simulation (which then would no longer be the bottleneck in the complexity).

B.2 Applications of smooth functions of Hamiltonians

In this subsection we use the input model for the d-sparse matrix H as described at the start of Section 2. We calculate the implementation cost in terms of queries to the input oracles (4)-(5), but it is easy to convert the results to more general statements as in the previous subsection.

The following theorem shows how to efficiently implement the function e^{-H} for some $H \succeq I$. We use this result in the proof of Lemma 13 to estimate expectation values of the quantum state $\rho = e^{-H}/\mathrm{Tr}\left(e^{-H}\right)$ (for the application we ensure that $H \succeq I$ by adding some multiple of I).

Theorem 43. *Suppose that $I \preceq H$ and we are given $K \in \mathbb{R}_+$ such that $\|H\| \leq 2K$. If $\varepsilon \in (0, 1/3)$, then we can implement a unitary $\tilde{U}$ such that $\left\|(\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - e^{-H}\right\| \leq \varepsilon$ using*

$$\mathcal{O}\left(Kd \log\left(\frac{K}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right)\right) \text{ queries and } \mathcal{O}\left(Kd \log\left(\frac{K}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right) \left[\log(n) + \log^{\frac{5}{2}}\left(\frac{K}{\varepsilon}\right)\right]\right) \text{ gates.}$$

Proof. In order to use Theorem 40 we set $x_0 := K + 1/2$ so that $\|H - x_0 I\| \leq K =: r$, and use the function

$$f(x_0 + x) = e^{-x_0 - x} = e^{-x_0} e^{-x} = e^{-x_0} \sum_{\ell=0}^{\infty} \frac{(-x)^\ell}{\ell!}.$$

We choose $\delta := 1/2$ so that $e^{-x_0} \sum_{\ell=0}^{\infty} \frac{(r+\delta)^\ell}{\ell!} = e^{-x_0} \sum_{\ell=0}^{\infty} \frac{x_0^\ell}{\ell!} = 1$, therefore we set $B := 1$. Theorem 40 tells us that we can implement a unitary $\tilde{U}$, such that $\tilde{f}(H) := (\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I)$ is an ε-approximation of $f(H)/B = e^{-H}$, using

$$\mathcal{O}\left(Kd \log\left(\frac{K}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right)\right) \text{ queries and } \mathcal{O}\left(Kd \log\left(\frac{K}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right) \left[\log(n) + \log^{\frac{5}{2}}\left(\frac{K}{\varepsilon}\right)\right]\right) \text{ gates.}$$

□

To conclude this appendix, we now sketch the proofs of a few interesting consequences of the earlier results in this appendix. These will, however, not be used in the body of the paper.

First, we show how to use the above subroutine together with amplitude amplification to prepare a Gibbs state with cost depending logarithmically on the precision parameter, as shown by the following lemma. To our knowledge this is the first Gibbs sampler that achieves

logarithmic dependence on the precision parameter without assuming access to the entries of $\sqrt{H}$ as in [CS17]. This can mean a significant reduction in complexity; for more details see the introduction of Section 2.2.3.

Lemma 44. *We can probabilistically prepare a purified Gibbs state $|\tilde{\gamma}\rangle_{AB}$ such that with high probability $\left\|\mathrm{Tr}_B(|\tilde{\gamma}\rangle\langle\tilde{\gamma}|_{AB}) - e^{-H}/\mathrm{Tr}\left(e^{-H}\right)\right\|_1 \leq \varepsilon$ holds, using an expected cost $\widetilde{\mathcal{O}}\left(\sqrt{n/\mathrm{Tr}(e^{-H})}\right)$ times the complexity of Theorem 43. If we are given a number $z \leq \mathrm{Tr}\left(e^{-H}\right)$, then we can also prepare $|\tilde{\gamma}\rangle_{AB}$ in a unitary fashion with cost $\widetilde{\mathcal{O}}\left(\sqrt{n/z}\right)$ times the complexity of Theorem 43.*

Sketch of proof. First we show how to prepare a purified sub-normalized Gibbs state. Then we use the exponential search algorithm of Boyer et al. [BBHT98] (with exponentially decreasing guesses for the norm a of the subnormalized Gibbs state, and hence exponentially increasing number of amplitude amplification steps) to postselect on this sub-normalized state in a similar fashion as in Algorithm 3. There is a possible caveat here: if we postselect on a state with norm a, then it gets rescaled by $1/a$ and its preparation error is rescaled by $1/a$ as well. Therefore during the rounds of the search algorithm we always increase the precision of implementation to compensate for the increased (error) amplification. Since the success of postselection in a round is upper bounded by the square of $\mathcal{O}(a \cdot \#\{\text{amplification steps in the round}\})$, the probability for the postselection to succeed in any of the early rounds is small.

Now we describe how to prepare a purified sub-normalized Gibbs state. We use the decomposition $H = \sum_{j=1}^{n} E_j |\phi_j\rangle\langle\phi_j|$, where $\{|\phi_j\rangle : j \in [n]\}$ is an orthonormal eigenbasis of H. Due to the invariance of maximally entangled states under transformations of the form $W \otimes W^*$ for unitary W, we have

$$\frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j\rangle_A |j\rangle_B = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |\phi_j\rangle_A |\phi_j^*\rangle_B. \tag{20}$$

Suppose we can implement a unitary U such that $(\langle 0| \otimes I)U(|0\rangle \otimes I) = e^{-H/2}$. If we apply U to the A-register of the state (20), then we get a state $|\gamma\rangle$ such that $\mathrm{Tr}_B((\langle 0| \otimes I)|\gamma\rangle\langle\gamma|(|0\rangle \otimes I)) = e^{-H}/n$.

If we implement $e^{-H/2}$ with sufficient precision using Theorem 43 in the exponential search algorithm, then after $\mathcal{O}\left(\sqrt{n/\mathrm{Tr}(e^{-H})}\right)$ rounds of amplitude amplification, with high probability we obtain a Gibbs state using the claimed expected runtime.

If we also know a lower bound $z \leq \mathrm{Tr}\left(e^{-H}\right)$, then we have an upper bound on the expected runtime, therefore we can turn the procedure into a unitary circuit using standard techniques. □

We can also recover the Gibbs sampler of Chowdhury and Somma [CS17]: if we apply our Corollary 42 to the function e^{-x^2} assuming access to $\sqrt{H}$ for some psd matrix H, then we get a Gibbs sampler for the state $e^{-H}/\mathrm{Tr}(e^{-H})$, similar to [CS17]. The advantage of the presented approach is that it avoids the usage of involved integral transformations, and can be presented without writing down a single integral sign, also due to our general results the proof is significantly shorter. Before we prove the precise statement in Lemma 46, we need some preparation for the application of Corollary 42:

Lemma 45. *For all $k \in \mathbb{N}$ we have*

$$\partial_x^{k+1} e^{-x^2} = -2x\partial_x^k e^{-x^2} - 2k\partial_x^{k-1} e^{-x^2} \tag{21}$$

and for all $x \in \mathbb{R}$

$$\left|\partial_x^k e^{-x^2}\right| \leq (2|x| + 2k)^k e^{-x^2}. \tag{22}$$

Therefore

$$\sum_{k=0}^{\infty} \frac{\left|\partial_x^k e^{-x^2}\right|}{k!}\left(\frac{1}{8e}\right)^k \leq 2. \tag{23}$$

Proof. We prove both claims by induction. $\partial_x^0 e^{-x^2} = e^{-x^2}$, $\partial_x^1 e^{-x^2} = -2xe^{-x^2}$ and $\partial_x^2 e^{-x^2} = 4x^2e^{-x^2} - 2e^{-x^2}$, so (21) holds for $k = 1$. Suppose (21) holds for k, we prove the inductive step as follows:

$$\partial_x^{k+2} e^{-x^2} = \partial_x\left(\partial_x^{k+1} e^{-x^2}\right) \overset{(21)}{=} \partial_x\left(-2x\partial_x^k e^{-x^2} - 2k\partial_x^{k-1} e^{-x^2}\right) = -2x\partial_x^{k+1} e^{-x^2} - 2(k+1)\partial_x^k e^{-x^2}.$$

Similarly, observe that (22) holds for $k = 0$ and $k = 1$. Suppose (22) holds for k, then we show the induction step as follows:

$$\begin{aligned}
\left|\partial_x^{k+1} e^{-x^2}\right| &\overset{(21)}{=} \left|-2x\partial_x^k e^{-x^2} - 2k\partial_x^{k-1} e^{-x^2}\right| \\
&\leq \left|2x\partial_x^k e^{-x^2}\right| + \left|2k\partial_x^{k-1} e^{-x^2}\right| \\
&\overset{(22)}{\leq} 2|x|(2|x| + 2k)^k e^{-x^2} + 2k(2|x| + 2(k-1))^{k-1} e^{-x^2} \\
&\leq (2|x| + 2(k+1))^{k+1} e^{-x^2}.
\end{aligned}$$

Finally, using the previous two statements we can prove (23) by the following calculation:

$$\begin{aligned}
\sum_{k=0}^{\infty} \frac{\left|\partial_x^k e^{-x^2}\right|}{k!}\left(\frac{1}{8e}\right)^k &\overset{(22)}{\leq} \sum_{k=0}^{\infty} \frac{(2|x| + 2k)^k e^{-x^2}}{k!}\left(\frac{1}{8e}\right)^k \\
&\leq \sum_{k=0}^{\infty} \frac{(4|x|)^k e^{-x^2}}{k!}\left(\frac{1}{8e}\right)^k + \sum_{k=1}^{\infty} \frac{(4k)^k e^{-x^2}}{k!}\left(\frac{1}{8e}\right)^k \\
&\leq e^{-x^2}\left(\sum_{k=0}^{\infty} \frac{1}{k!}\left(\frac{4|x|}{8e}\right)^k + \sum_{k=1}^{\infty} \frac{1}{k!}\left(\frac{4k}{8e}\right)^k\right) \\
&\leq e^{-x^2}\left(e^{\frac{|x|}{2e}} + \sum_{k=1}^{\infty} \frac{1}{\sqrt{2\pi}}\left(\frac{e}{k}\right)^k\left(\frac{k}{2e}\right)^k\right) \\
&= e^{-\left(|x| - \frac{1}{4e}\right)^2} e^{\left(\frac{1}{4e}\right)^2} + \frac{e^{-x^2}}{\sqrt{2\pi}} \\
&\leq e^{\left(\frac{1}{4e}\right)^2} + \frac{1}{\sqrt{2\pi}} \\
&\leq 2.
\end{aligned}$$

□

Lemma 46. *Suppose we know a $K > 1$ such that $\|H\| \leq K$. If $\varepsilon \in (0, 1/3)$, then we can implement a unitary $\tilde{U}$ such that $\left\|(\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - e^{-H^2}/2\right\| \leq \varepsilon$ using*

$$\mathcal{O}\left(Kd\log\left(\frac{K}{\varepsilon}\right)\log\left(\frac{1}{\varepsilon}\right)\right) \text{ queries and } \mathcal{O}\left(Kd\log\left(\frac{K}{\varepsilon}\right)\log\left(\frac{1}{\varepsilon}\right)\left[\log(n) + \log^{\frac{5}{2}}\left(\frac{K}{\varepsilon}\right)\right]\right) \text{ gates.}$$

Proof. We apply Corollary 42 to the function e^{-x^2}. For this let $L_0 := \lceil 32K \rceil$, $L := 2L_0 + 1$ and let $x_\ell := (\ell - 1 - L_0)/32$ for all $\ell \in [L]$. We choose $r := 1/32$, $\delta := \theta := 1/128$ and $B = 2$ so that the conditions of Corollary 42 are satisfied, as shown by Lemma 45. Indeed $r + \delta \leq 1/(8e)$, hence for all $\ell \in [L]$ we have $\sum_{k=0}^{\infty} a_k^{(\ell)}(r+\delta)^k \leq 2 = B$ as we can see by (23). Since $\delta = \Theta(1)$, Corollary 42 provides the desired complexity. □

We can use the above lemma to prepare Gibbs states in a similar way to Lemma 44. In case we have access to $\sqrt{H}$, the advantage of this method is that the dependence on $\|H\|$ is reduced to $\sqrt{\|H\|}$.

Improved HHL algorithm. Our techniques can also be applied to the Harrow-Hassidim-Lloyd (HHL) algorithm [HHL09] in order to gain improvements in a similar manner to Childs et al. [CKS17]. The problem the HHL algorithm solves is the following. Suppose we have a circuit U preparing a quantum state $|b\rangle$ (say, starting from $|0\rangle$), and have d-sparse oracle access to a non-singular Hamiltonian H. The task is to prepare a quantum state, that ε-approximates $H^{-1}|b\rangle/\|H^{-1}|b\rangle\|$. For simplicity here we only count the number of uses of U and the number of queries to H. Childs et al. [CKS17] present two different methods for achieving this, one based on Hamiltonian simulation, and another directly based on quantum walks. Under the conditions $\|H\| \leq 1$ and $\|H^{-1}\| \leq \kappa$, the former makes $\mathcal{O}\left(\kappa\sqrt{\log(\kappa/\varepsilon)}\right)$ uses of U and has query complexity $\mathcal{O}(d\kappa^2 \log^{2.5}(\kappa/\varepsilon))$. The latter makes $\mathcal{O}(\kappa \log(d\kappa/\varepsilon))$ uses of U and has query complexity $\mathcal{O}(d\kappa^2 \log^2(d\kappa/\varepsilon))$.

Now we provide a sketch of how to solve the HHL problem with $\mathcal{O}(\kappa)$ uses of U and with query complexity $\mathcal{O}(d\kappa^2 \log^2(\kappa/\varepsilon))$ using our techniques. The improvement on both previously mentioned results is not very large, but our proof is significantly shorter thanks to our general Theorem 40 and Corollary 42.

To solve the HHL problem we need to implement the function H^{-1}, i.e., apply the function $f(x) = 1/x$ to H. Due to the constraints on H, the eigenvalues of H lie in the union of the intervals $[-1, -1/\kappa]$ and $[1/\kappa, 1]$. We first assume that the eigenvalues actually lie in $[1/\kappa, 1]$. In this case we can easily implement the function $1/x$ by Theorem 40 using the Taylor series around 1:

$$(1+z)^{-1} = \frac{1}{1+z} = \sum_{k=0}^{\infty} (-1)^k z^k. \tag{24}$$

As $H^{-1} = (I + (H - I))^{-1}$, we are interested in the eigenvalues of $H - I$. The eigenvalues of $H - I$ lie in the interval $[-1+1/\kappa, 0]$, so we choose $r := 1 - 1/\kappa$ and $\delta := 1/(2\kappa)$. By substituting $z := -1 + 1/(2\kappa)$ in (24), we can see that $B := 2\kappa$ satisfies the conditions of Theorem 40. Let $\varepsilon' \in (0, 1/2)$, then Theorem 40 provides an $\mathcal{O}(d\kappa \log(\kappa/\varepsilon') \log(1/\varepsilon'))$-query implementation of an ε'-approximation of the operator $H^{-1}/(2\kappa)$, since $\|H\| \leq 1$. We can proceed similarly when the eigenvalues of $H - I$ lie in the interval $[-1, -1/\kappa]$, and we can combine the two cases using Corollary 42.

Setting $\varepsilon' := c\varepsilon/\kappa$ for an appropriate constant c and using amplitude amplification, we can prepare an ε-approximation of the state $H^{-1}|b\rangle/\|H^{-1}|b\rangle\|$ as required by HHL using $\mathcal{O}(\kappa)$ amplitude amplification steps. Therefore we use U at most $\mathcal{O}(\kappa)$ times and make $\mathcal{O}\left(d\kappa^2 \log^2(\kappa/\varepsilon)\right)$ queries.

C Generalized minimum-finding algorithm

In this appendix we describe our generalized quantum minimum-finding algorithm, which we are going to apply to finding an approximation of the ground state energy of a Hamiltonian. This algorithm generalizes the results of Dürr and Høyer [DH96] in a manner similar to the way amplitude amplification [BHMT02] generalizes Grover search: we do not need to assume the ability to query individual elements of the search space, we just need to be able to generate a superposition over the search space. The algorithm also has the benefit over binary search that it removes a logarithmic factor from the complexity.

The backbone of our analysis will be the meta-algorithm below from [DH96]. The meta-algorithm finds the minimal element in the range of the random variable X by sampling, where by "range" we mean the values which occur with non-zero probability. We assume X has finite range.

Input A discrete random variable X with finite range.

Output The minimal value $x_{\min}$ in the range of X.

Init $t \leftarrow 0$; $s_0 \leftarrow \infty$
Repeat until s_t is minimal in the range of X

1. $t \leftarrow t + 1$
2. Sample a value s_t according to the conditional distribution $\Pr(X = s_t \mid X < s_{t-1})$.

Meta-Algorithm 2: Minimum-finding

Note that the above algorithm will always find the minimum, since the obtained samples are strictly decreasing.

Lemma 47. *Let X be a finite discrete random variable whose range of values is $x_1 < x_2 < \ldots < x_N$. Let $S(X) = \{s_1, s_2, \ldots\}$ denote the random set of values obtained via sampling during a run of Meta-Algorithm 2 with input random variable X. If $k \in [N]$, then*

$$\Pr(x_k \in S(X)) = \frac{\Pr(X = x_k)}{\Pr(X \le x_k)}.$$

Proof. The intuition of the proof is to show that, whenever $\Pr(s_{t-1} > x_k) > 0$, we have

$$\Pr(s_t = x_k \mid t \in [N] \text{ is the first time such that } s_t \le x_k) = \frac{\Pr(X = x_k)}{\Pr(X \le x_k)}. \tag{25}$$

To formulate the statement more precisely[13] we consider a fixed value $t \in [N]$. For notational convenience let $x := x_k$, $x_{N+1} := \infty$, we prove (25) by:

$$\begin{aligned}
\Pr(s_t = x \mid s_t \le x \wedge s_{t-1} > x) &= \frac{\Pr(s_t = x)}{\Pr(s_t \le x \wedge s_{t-1} > x)} \\
&= \sum_{x_\ell > x} \frac{\Pr(s_t = x \wedge s_{t-1} = x_\ell)}{\Pr(s_t \le x \wedge s_{t-1} > x)} \\
&= \sum_{x_\ell > x} \frac{\Pr(s_t = x \wedge s_{t-1} = x_\ell)}{\Pr(s_t \le x \wedge s_{t-1} = x_\ell)} \frac{\Pr(s_t \le x \wedge s_{t-1} = x_\ell)}{\Pr(s_t \le x \wedge s_{t-1} > x)} \\
&= \sum_{x_\ell > x} \frac{\Pr(s_t = x \mid s_{t-1} = x_\ell)\Pr(s_{t-1} = x_\ell)}{\Pr(s_t \le x \mid s_{t-1} = x_\ell)\Pr(s_{t-1} = x_\ell)} \frac{\Pr(s_t \le x \wedge s_{t-1} = x_\ell)}{\Pr(s_t \le x \wedge s_{t-1} > x)} \\
&= \sum_{x_\ell > x} \frac{\Pr(X = x)}{\Pr(X \le x)} \frac{\Pr(s_{t-1} = x_\ell)}{\Pr(s_{t-1} = x_\ell)} \frac{\Pr(s_t \le x \wedge s_{t-1} = x_\ell)}{\Pr(s_t \le x \wedge s_{t-1} > x)} \\
&= \frac{\Pr(X = x)}{\Pr(X \le x)} \frac{\Pr(s_{t-1} > x)}{\Pr(s_{t-1} > x)}.
\end{aligned}$$

[13] Throughout this proof, whenever a fraction is $0/0$, we simply interpret it as 0. Therefore we also interpret conditional probabilities, conditioned on events that happen with probability 0, as 0.

This is enough to conclude the proof, since there is always a smallest $t \in [N]$ such that $s_t \leq x$, as the algorithm always finds the minimum in at most N steps. So we can finish the proof by

$$\begin{aligned}
\Pr(x \in S(X)) &= \sum_{t=1}^{N} \Pr(s_t = x) \\
&= \sum_{t=1}^{N} \Pr(s_t = x \mid s_t \leq x \wedge s_{t-1} > x) \Pr(s_t \leq x \wedge s_{t-1} > x) \\
&= \frac{\Pr(X = x)}{\Pr(X \leq x)} \sum_{t=1}^{N} \Pr(s_t \leq x \wedge s_{t-1} > x) \\
&= \frac{\Pr(X = x)}{\Pr(X \leq x)} \Pr(\exists t \in [N] \colon s_t \leq x \wedge s_{t-1} > x) \\
&= \frac{\Pr(X = x)}{\Pr(X \leq x)}.
\end{aligned}$$

□

Now we describe our generalized minimum-finding algorithm which is based on Meta-Algorithm 2. We take some unitary U, and replace X by the distribution obtained if we measured the second register of $U|0\rangle$. We implement conditional sampling via amplitude amplification and the use of the exponential search algorithm of Boyer et al. [BBHT98]. If a unitary prepares the state $|0\rangle|\phi\rangle + |1\rangle|\psi\rangle$ where $\|\phi\|^2 + \|\psi\|^2 = 1$, then this exponential search algorithm built on top of amplitude amplification prepares the state $|1\rangle|\psi\rangle$ probabilistically using an expected number of $\mathcal{O}(1/\|\psi\|)$ applications of U and U^{-1} (we will skip the details here, which are straightforward modifications of [BBHT98]).

Input A number M and a unitary U, acting on q qubits, such that $U|0\rangle = \sum_{k=1}^{N} |\psi_k\rangle|x_k\rangle$, where x_k is a binary string representing some number and $|\psi_k\rangle$ is an unnormalized quantum state on the first register. Let $x_1 < x_2 < \ldots < x_N$ and define X to be the random variable with $\Pr(X = x_k) = \|\psi_k\|^2$.

Output Some $|\psi_k\rangle|x_k\rangle$ for a (hopefully) small k.

Init $t \leftarrow 0$; $s_0 \leftarrow \infty$

While the total number of applications of U and U^{-1} does not exceed M:

1. $t \leftarrow t + 1$
2. Use the exponential search algorithm with amplitude amplification on states such that $x_k < s_{t-1}$ to obtain a sample $|\psi_k\rangle|x_k\rangle$.
3. $s_t \leftarrow x_k$

Algorithm 3: Generalized minimum-finding

Lemma 48. *There exists $C \in \mathbb{R}_+$, such that if we run Algorithm 3 indefinitely (setting $M = \infty$), then for every U and x_k the expected number of uses of U and U^{-1} before obtaining a sample $x \leq x_k$ is at most $\frac{C}{\sqrt{\Pr(X \leq x_k)}}$.*

Proof. Let $X_{<x_\ell}$ denote the random variable for which $\Pr(X_{<x_\ell} = x) = \Pr(X = x \mid X < x_\ell)$.

The expected number of uses of $U^{\pm 1}$ in Algorithm 3 before obtaining a value $x \le x_k$ is

$$\begin{aligned}
\mathbb{E}\Big[\#\text{uses of } U^{\pm 1} \text{ for finding } x \le x_k\Big] &= \sum_{i=0}^{N-1} \mathbb{E}\Big[\#\text{uses of } U^{\pm 1} \text{ in the } i\text{-th round before } x \le x_k\Big] \\
&= 1 + \sum_{i=1}^{N-1} \sum_{\ell=k+1}^{N} \Pr(s_i = x_\ell)\, \mathbb{E}\Big[\#\text{uses of } U^{\pm 1} \text{ for sampling } X_{<x_\ell}\Big] \\
&= 1 + \sum_{\ell=k+1}^{N} \Pr(x_\ell \in S(X))\, \mathbb{E}\Big[\#\text{uses of } U^{\pm 1} \text{ for sampling } X_{<x_\ell}\Big] \\
&= 1 + \sum_{\ell=k+1}^{N} \frac{\Pr(X = x_\ell)}{\Pr(X \le x_\ell)} \mathcal{O}\left(\frac{1}{\sqrt{\Pr(X < x_\ell)}}\right) \\
&= \mathcal{O}\left(\sum_{\ell=k+1}^{N} \frac{\Pr(X = x_\ell)}{\Pr(X \le x_\ell)} \frac{1}{\sqrt{\Pr(X < x_\ell)}}\right) \\
&= \mathcal{O}\left(\frac{1}{\sqrt{\Pr(X \le x_k)}}\right),
\end{aligned}$$

where the last equality follows from Equation (30) below. The constant C from the lemma is the constant hidden by the $\mathcal{O}$. The remainder of this proof consists of proving (30) using elementary calculus. Let us introduce the notation $p_0 := \Pr(X \le x_k)$ and for all $j \in [N-k]$ let $p_j := \Pr(X = x_{k+j})$. Then the expression inside the $\mathcal{O}$ on the second-to-last line above becomes

$$\sum_{\ell=k+1}^{N} \frac{\Pr(X = x_\ell)}{\Pr(X \le x_\ell)} \sqrt{\frac{1}{\Pr(X < x_\ell)}} = \sum_{j=1}^{N-k} \frac{p_j}{\sum_{i=0}^{j} p_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} p_i}}. \tag{26}$$

The basic idea is that we treat the expression on the right-hand side of (26) as an integral approximation sum for the integral $\int_{p_0}^{1} z^{-3/2} dz$, and show that it is actually always less than the value of this integral. We proceed by showing that subdivision always increases the sum.

Let us fix some $\ell \in [N-k]$ and define

$$p'_i = \begin{cases} p_i & \text{for } i \in \{0, 1, \ldots, \ell - 1\} \\ p_i/2 & \text{for } i \in \{\ell, \ell + 1\} \\ p_{i-1} & \text{for } i \in \{\ell + 2, \ldots, N - k + 1\} \end{cases}$$

and observe that

$$\begin{aligned}
&\sum_{j=1}^{N-k+1} \frac{p'_j}{\sum_{i=0}^{j} p'_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} p'_i}} - \sum_{j=1}^{N-k} \frac{p_j}{\sum_{i=0}^{j} p_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} p_i}} \\
&= \frac{p'_\ell}{\sum_{i=0}^{\ell} p'_i} \sqrt{\frac{1}{\sum_{i=0}^{\ell-1} p'_i}} + \frac{p'_{\ell+1}}{\sum_{i=0}^{\ell+1} p'_i} \sqrt{\frac{1}{\sum_{i=0}^{\ell} p'_i}} - \frac{p_\ell}{\sum_{i=0}^{\ell} p_i} \sqrt{\frac{1}{\sum_{i=0}^{\ell-1} p_i}} \\
&= \frac{p_\ell/2}{p_\ell/2 + \sum_{i=0}^{\ell-1} p_i} \sqrt{\frac{1}{\sum_{i=0}^{\ell-1} p_i}} + \frac{p_\ell/2}{p_\ell + \sum_{i=0}^{\ell-1} p_i} \sqrt{\frac{1}{p_\ell/2 + \sum_{i=0}^{\ell-1} p_i}} - \frac{p_\ell}{p_\ell + \sum_{i=0}^{\ell-1} p_i} \sqrt{\frac{1}{\sum_{i=0}^{\ell-1} p_i}}.
\end{aligned} \tag{27}$$

We show that (27) is ≥ 0 after simplifying the expression by substituting $a := \sum_{i=0}^{\ell-1} p_i$ and $b := p_\ell/2$:

$$\frac{b}{a+b}\sqrt{\frac{1}{a}} + \frac{b}{a+2b}\sqrt{\frac{1}{a+b}} - \frac{2b}{a+2b}\sqrt{\frac{1}{a}} = \frac{\Big(a + b - \sqrt{a}\sqrt{a+b}\Big)b}{(a+b)^{3/2}(a+2b)} \ge 0. \tag{28}$$

Let us fix some parameter $\delta > 0$. Recursively applying this halving procedure for different indices, we can find some $J \in \mathbb{N}$ and $\tilde{p} \in \mathbb{R}_+^{J+1}$ such that $\sum_{j=0}^{J} \tilde{p}_j = 1$, $\tilde{p}_0 = p_0$ and $\tilde{p}_j \leq \delta$ for all $j \in [J]$, moreover

$$\sum_{j=1}^{N-k} \frac{p_j}{\sum_{i=0}^{j} p_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} p_i}} \leq \sum_{j=1}^{J} \frac{\tilde{p}_j}{\sum_{i=0}^{j} \tilde{p}_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} \tilde{p}_i}}.$$

Observe that for all $j \in [J]$

$$\begin{aligned}\sqrt{\frac{1}{\sum_{i=0}^{j-1} \tilde{p}_i}} = \sqrt{\frac{1}{\sum_{i=0}^{j} \tilde{p}_i}} \sqrt{\frac{\sum_{i=0}^{j} \tilde{p}_i}{\sum_{i=0}^{j-1} \tilde{p}_i}} &= \sqrt{\frac{1}{\sum_{i=0}^{j} \tilde{p}_i}} \sqrt{1 + \frac{\tilde{p}_j}{\sum_{i=0}^{j-1} \tilde{p}_i}} \\ &\leq \sqrt{\frac{1}{\sum_{i=0}^{j} \tilde{p}_i}} \sqrt{1 + \frac{\delta}{\tilde{p}_0}} \leq \sqrt{\frac{1}{\sum_{i=0}^{j} \tilde{p}_i}} \left(1 + \frac{\delta}{p_0}\right). \end{aligned} \tag{29}$$

Therefore

$$\begin{aligned}\sum_{j=1}^{N-k} \frac{p_j}{\sum_{i=0}^{j} p_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} p_i}} &\leq \sum_{j=1}^{J} \frac{\tilde{p}_j}{\sum_{i=0}^{j} \tilde{p}_i} \sqrt{\frac{1}{\sum_{i=0}^{j-1} \tilde{p}_i}} \\ \text{(by (29))} \quad &\leq \sum_{j=1}^{J} \frac{\tilde{p}_j}{\sum_{i=0}^{j} \tilde{p}_i} \sqrt{\frac{1}{\sum_{i=0}^{j} \tilde{p}_i}} \left(1 + \frac{\delta}{p_0}\right) \\ &= \left(1 + \frac{\delta}{p_0}\right) \sum_{j=1}^{J} \frac{\tilde{p}_j}{\left(\sum_{i=0}^{j} \tilde{p}_i\right)^{3/2}} \\ &= \left(1 + \frac{\delta}{p_0}\right) \sum_{j=1}^{J} \int_{\sum_{i=0}^{j-1} \tilde{p}_i}^{\sum_{i=0}^{j} \tilde{p}_i} \frac{1}{\left(\sum_{i=0}^{j} \tilde{p}_i\right)^{3/2}} dz \\ &\leq \left(1 + \frac{\delta}{p_0}\right) \sum_{j=1}^{J} \int_{\sum_{i=0}^{j-1} \tilde{p}_i}^{\sum_{i=0}^{j} \tilde{p}_i} z^{-\frac{3}{2}} dz \\ &= \left(1 + \frac{\delta}{p_0}\right) \int_{p_0}^{1} z^{-\frac{3}{2}} dz \\ &= \left(1 + \frac{\delta}{p_0}\right) \left[-2z^{-\frac{1}{2}}\right]_{p_0}^{1} \\ &\leq \left(1 + \frac{\delta}{p_0}\right) \left(\frac{2}{\sqrt{p_0}}\right). \end{aligned}$$

Since this inequality holds for every $\delta > 0$, we can conclude using (26) that

$$\sum_{\ell=k+1}^{N} \frac{\Pr(X = x_\ell)}{\Pr(X \leq x_\ell)} \sqrt{\frac{1}{\Pr(X < x_\ell)}} \leq \frac{2}{\sqrt{p_0}} = \frac{2}{\sqrt{\Pr(X \leq x_k)}}. \tag{30}$$

□

It is not too hard to work out the constant by following the proof of [BBHT98] providing something like $C \approx 25$. The following theorem works with any C satisfying Lemma 48.

Theorem 49 (Generalized Minimum-Finding)**.** *If we run Algorithm 3 with input satisfying $M \geq 4C/\sqrt{\Pr(X \leq x)}$ for C as in Lemma 48 and a unitary U that acts on q qubits, then at termination we obtain an x_i from the range of X that satisfies $x_i \leq x$ with probability at least $\frac{3}{4}$. Moreover the success probability can be boosted to at least $1 - \delta$ with $\mathcal{O}(\log(1/\delta))$ repetitions. This uses at most M applications of U and U^{-1} and $\mathcal{O}(qM)$ other gates.*

Proof. Let x_k be the largest value in the range of X such that $x_k \leq x$. Then Lemma 48 says that the expected number of applications of U and U^{-1} before finding a value $x_i \leq x_k$ is at most $C/\sqrt{\Pr(X \leq x_k)} = C/\sqrt{\Pr(X \leq x)}$, therefore by the Markov inequality we know that the probability that we need to use U and U^{-1} at least $4C/\sqrt{\Pr(X \leq x)}$ times is at most $1/4$. The boosting of the success probability can be done using standard techniques, e.g., by repeating the whole procedure $\mathcal{O}(\log(1/\delta))$ times and taking the minimum of the outputs.

The number of applications of U and U^{-1} follows directly form the algorithms description. Then, for the number of other gates, each amplitude amplification step needs to implement a binary comparison and a reflection through the $|0\rangle$ state, both of which can be constructed using $\mathcal{O}(q)$ elementary gates, giving a total of $\mathcal{O}(qM)$ gates. □

Note that this result is a generalization of Dürr and Høyer [DH96]: if we can create a uniform superposition over N values $x_1 < x_2 < \ldots < x_N$, then $\Pr(X \leq x_1) = 1/N$ and therefore Theorem 49 guarantees that we can find the minimum with high probability with $\mathcal{O}(\sqrt{N})$ steps.

Now we describe an application of this generalized search algorithm that we need in the paper.

This final lemma in this appendix describes how to estimate the smallest eigenvalue of a Hamiltonian. A similar result was shown by Poulin and Wocjan [PW09a], but we improve on the analysis to fit our framework better. We assume sparse oracle access to the Hamiltonian H as described in Section 2, and will count queries to these oracles. We use some of the techniques introduced in Appendix B.

Lemma 50. *If $H = \sum_{j=1}^{n} E_j |\phi_j\rangle\langle\phi_j|$, with eigenvalues $E_1 \leq E_2 \leq \ldots \leq E_n$, is such that $\|H\| \leq K$, $\varepsilon \leq K/2$, and H is given in d-sparse oracle form, then we can obtain an estimate E such that $|E_1 - E| \leq \varepsilon$, with probability at least $2/3$, using*

$$\mathcal{O}\left(\frac{Kd\sqrt{n}}{\varepsilon}\log^2\left(\frac{Kn}{\varepsilon}\right)\right) \text{ queries and } \mathcal{O}\left(\frac{Kd\sqrt{n}}{\varepsilon}\log^{\frac{9}{2}}\left(\frac{Kn}{\varepsilon}\right)\right) \text{ gates.}$$

Proof. The general idea is as follows: we prepare a maximally entangled state on two registers, and apply phase estimation [NC00, Section 5.2][CEMM98] to the first register with respect to the unitary $e^{\pi i H/K}$. We then use Theorem 49 to find the minimal phase. In order to guarantee correctness we need to account for all the approximation errors coming from approximate implementations. This causes some technical difficulty, since the approximation errors can introduce phase estimates that are much less than the true minimum. We need to make sure that the minimum-finding algorithm finds these faulty estimates only with a tiny probability.

We first initialize two $\log(n)$-qubit registers in a maximally entangled state $\frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}|j\rangle|j\rangle$. This can be done for example using $\log(n)$ Hadamard and CNOT gates, when n is a power of two. Due to the invariance of maximally entangled states under transformations of the form $W \otimes W^*$ for unitary W, we have that

$$\frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}|j\rangle|j\rangle = \frac{1}{\sqrt{n}}\sum_{j=1}^{n}|\phi_j\rangle|\phi_j^*\rangle.$$

Let $T := 2^{\lceil \log(\frac{K}{\varepsilon})+2 \rceil}$ and first assume that we have access to a perfect unitary V which implements $V = \sum_{t=0}^{T-1}|t\rangle\langle t| \otimes e^{\pi t i H/K}$. Let $e_j := E_j T/2K$. If we apply phase estimation to the quantum state $|\phi_j\rangle$ using V, then we get some superposition of phase estimates $|e\rangle$ in the first register. This superposition has the property, that a measurement in the computational basis reveals an e such that $|e - e_j| \leq 3$ with high probability, so that the estimate $E := e2K/T$

would satisfy $|E - E_j| = |e - e_j|2K/T \leq 3\varepsilon/4 < \varepsilon$. If we repeat phase estimation $\mathcal{O}(\log(n))$ times (on the same state $|\phi_j\rangle$), and take the median of the estimates (in superposition), then we obtain a more concentrated superposition of estimates e such that a measurement would reveal an $|e - e_j| \leq 3$ with probability at least $1 - b/n$, for some $b = \Theta(1)$.

Since in our maximally entangled state $|\phi_j\rangle$ is entangled with $|\phi_j^*\rangle$ on the second register, applying phase estimation to the first register in superposition does not cause interference. Let us denote by U the unitary corresponding to the above preparation-estimation-boost procedure. Define Π to be the projector which projects to the subspace of estimation values $|e\rangle$ such that there is a $j \in [n]$ with $|e - e_j| \leq \varepsilon$. By the non-interference argument we can see that, after applying U, the probability that we get an estimation e such that $|e - e_j| > 3$ is at most b/n for all $j \in [n]$, moreover $\|(I - \Pi)U|0\rangle\|^2 \leq b/n$. Also let Π_1 denote the projector which projects to phase estimates that yield e such that $|e - e_1| \leq 3$. It is easy to see that $\|\Pi_1 U|0\rangle\|^2 \geq 1/n - b/n^2$.

Now let us replace V by $\tilde{V}$ implemented via Lemma 36, such that $\|V - \tilde{V}\| \leq c'/(n\log(n))$ for some $c' = \Theta(1)$. Let $\tilde{U}$ denote the circuit that we obtain from U by replacing V with $\tilde{V}$. Since in the repeated phase-estimation procedure we use V in total $\mathcal{O}(\log(n))$ times, by using the triangle inequality we see that $\|U - \tilde{U}\| \leq c/(2n)$, where $c = \Theta(1)$. We use the well-known fact that if two unitaries are δ-close in operator norm, and they are applied to the same quantum state, then the measurement statistics of the resulting states are 2δ-close. Therefore we can upper bound the difference in probability of getting outcome $(I - \Pi)$:

$$\|(I - \Pi)\tilde{U}|0\rangle\|^2 - \|(I - \Pi)U|0\rangle\|^2 \leq 2\|U|0\rangle - \tilde{U}|0\rangle\| \leq c/n,$$

hence $\|(I - \Pi)\tilde{U}|0\rangle\|^2 \leq (b+c)/n$, and we can prove similarly that $\|\Pi_1\tilde{U}|0\rangle\|^2 \geq 1/n - (b+c)/n$.

Now let $|\psi\rangle := (I - \Pi)\tilde{U}|0\rangle/\|(I - \Pi)\tilde{U}|0\rangle\|$ be the state that we would get after post-selecting on the $(I - \Pi)$-outcome of the projective measurement Π. For small enough b, c we have that $\||\psi\rangle - \tilde{U}|0\rangle\| = \mathcal{O}(\sqrt{(b+c)/n})$ by the triangle inequality. Thus there exists an idealized unitary U' such that $|\psi\rangle = U'|0\rangle$, and $\|\tilde{U} - U'\| = \mathcal{O}(\sqrt{(b+c)/n})$. Observe that $\|\Pi_1 U'|0\rangle\|^2 = \||\psi\rangle\|^2 \geq \|\Pi_1\tilde{U}|0\rangle\|^2 \geq 1/n - (b+c)/n$.

Now suppose $(b + c) \leq 1/2$ and we run the generalized minimum-finding algorithm of Theorem 49 using U' with $M = 6C\sqrt{n}$. Since

$$\Pr(e \leq e_1 + 3) \geq \|\Pi_1 U'|0\rangle\|^2 \geq (1 - b - c)/n \geq 1/(2n) > 4/(9n)$$

we will obtain an estimate e such that $e \leq e_1 + 3$, with probability at least $3/4$. But since $\Pi|\psi\rangle = |\psi\rangle$, we find that any estimate that we might obtain satisfies $e \geq e_1 - 3$. So an estimate $e \leq e_1 + 3$ always satisfies $|e - e_1| \leq 3$.

The problem is that we only have access to $\tilde{U}$ as a quantum circuit. Let $C_{MF}(\tilde{U})$ denote the circuit that we get from Theorem 49 when using it with $\tilde{U}$ and define similarly $C_{MF}(U')$ for U'. Since we use $\tilde{U}$ a total of $\mathcal{O}(\sqrt{n})$ times in $C_{MF}(\tilde{U})$ and

$$\left\|\tilde{U} - U'\right\| = \mathcal{O}(\sqrt{(b+c)/n}), \text{ we get that } \left\|C_{MF}(\tilde{U}) - C_{MF}(U')\right\| = \mathcal{O}(\sqrt{b+c}).$$

Therefore the measurement statistics of the two circuits differ by at most $\mathcal{O}(\sqrt{b+c})$. Choosing b, c small enough constants ensures that $C_{MF}(\tilde{U})$ outputs a proper estimate e such that $|e - e_1| \leq 3$ with probability at least $2/3$. As we have shown at the beginning of the proof, such an e yields an ε-approximation of E_1 via $E := e2K/T$.

The query complexity has an $\mathcal{O}(Td\log(Tn)) = \mathcal{O}(Kd/\varepsilon\log(Kn/\varepsilon))$ factor coming from the implementation of $\tilde{V}$ by Lemma 36. This gets multiplied with $\mathcal{O}(\log(n))$ by the boosting of phase estimation, and by $\mathcal{O}(\sqrt{n})$ due to the minimum-finding algorithm. The gate complexity is dominated by the cost $\mathcal{O}(Kd/\varepsilon\log^{7/2}(Kn/\varepsilon))$ of implementing $\tilde{V}$, multiplied with the $\mathcal{O}(\sqrt{n}\log(n))$ factor as for the query complexity. □

Note that the minimum-finding algorithm of Theorem 49 can also be used for state preparation. If we choose 2ε less than the energy-gap of the Hamiltonian, then upon finding the approximation of the ground state energy we also prepare an approximate ground state. The precision of this state preparation can be improved with logarithmic cost, as can be seen from the proof of Lemma 50.

D Sparse matrix summation

As seen in Section 2, the Arora-Kale algorithm requires an approximation of $\exp(-\eta H^{(t)})$ where $H^{(t)}$ is a sum of matrices. To keep this section general we simplify the notation. Let H be the sum of k different d-sparse matrices M:

$$H = \sum_{i=1}^{k} M_i$$

In this section we study the complexity of one oracle call to H, given access to respective oracles for the matrices $M_1, \ldots, M_k$. Here we assume that the oracles for the M_i are given in sparse matrix form, as defined in Section 2. In particular, the goal is to construct a procedure that acts as a similar sparse matrix oracle for H. 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 separate oracles.

In the remainder of this section we only consider one row of H. We denote this row by R_H and the corresponding rows of the matrices M_i by R_i. Notice that such a row is given as an ordered list of integers, where the integers are the non-zero indices in R_i. Then 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).

D.1 A lower bound

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

Lemma 51. *Given $k+1$ ordered lists of integers $R_0, \ldots, R_k$, each of length at most d. 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 $|R_H| \leq d + \frac{dk}{2}$ or $|R_H| \geq d + \frac{dk}{2} + 1$ takes $\Omega(dk)$ quantum queries to the input lists in general.*

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

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

By construction, if $Z_{ij} = 1$, then the value of the entry $R_i[j]$ is unique in the lists $R_0, \ldots, R_k$, 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 one. The length of R_H is therefore $d + |Z|$. Hence, distinguishing between $|R_H| \leq d + \frac{dk}{2}$ and $|R_H| \geq d + \frac{dk}{2} + 1$ would solve the MAJ problem and therefore requires at least $\Omega(dk)$ queries to the lists in general. □

Corollary 52. *Implementing a query to a sparse matrix oracle O_H^I for*

$$H = \sum_{j=i}^{k} M_j$$

where each M_j is d-sparse, requires $\Omega(dk)$ queries to the $O_{M_j}^I$ in general.

D.2 An upper bound

We first show that an oracle for the non-zero indices of 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 H as follows. We start from the oracle of M_1 and then we "add" the oracle of M_2, then that of M_2, etc. By "adding" the oracle M_i to $H = \sum_{j=1}^{i-1} M_j$, we mean that, per row, we insert the non-zero indices in the list of M_i into that of H (if it is not already there). When an efficient data structure (for example a binary heap) is used, then such insertions can be done in polylog time. This shows that in the classical case such an oracle can be made in time $\widetilde{\mathcal{O}}(ndk)$. Note that in the application we are interested in, Meta-algorithm 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 $\widetilde{\mathcal{O}}(nd)$.

The quantum case is similar, but we need to add all the matrices together each time a query to O_H^I is made, since writing down each row of H in every iteration would take $\Omega(n)$ operations.

To implement one such query to O_H^I, in particular to the tth entry of R_H, start with an empty heap and add all elements of R_1 to it. Continue with the elements of R_2, but this time, for each element first check if it is already present, if not, add it, if it is, just continue. Overall this will take $\mathcal{O}(dk)$ insertions and searches in the data structure and hence $\widetilde{\mathcal{O}}(dk)$ operations. We end up with a full description of R_H. We can then find the index of the tth non-zero element from this and uncompute the whole description. Similarly, we need to be able to compute the inverse function since we need an in-place calculation. Given an index i of a non-zero element in R_H, we can compute all the indices for R_H as above, and find where i is in the heap to find the corresponding value of t.

E Equivalence of R, r, and ε^{-1}

In this section we will prove the equivalence of the three parameters R, r and ε^{-1} in the Arora-Kale meta-algorithm. 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 53. *Let $0 < \varepsilon \leq 1$. For every SDP with $R, r \geq 1$, there is an SDP with parameters $\tilde{R} = 1$ and $\tilde{r} = r$ such that a solution to that SDP with precision $\tilde{\varepsilon} = \frac{\varepsilon}{R}$ provides a solution with precision ε to 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 54. *Let $0 < \varepsilon \leq 1$. For every SDP with $R, r \geq 1$, there is an SDP with parameters $\tilde{R} = \frac{R}{\varepsilon}$ and $\tilde{r} = r$ such that a solution to that SDP with precision $\tilde{\varepsilon} = 1$ provides a solution with precision ε to 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 55. *Let $0 < \varepsilon \leq 1$. For every SDP with $R, r \geq 1$, there is an SDP with parameters $\tilde{R} = R$ and $\tilde{r} = 1$ such that a solution to that SDP with precision $\tilde{\varepsilon} = \frac{\varepsilon}{r}$ provides a solution with precision ε to 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 last reduction by setting $\tilde{C} = \frac{1}{\varepsilon}C$, but this would not guarantee that $\|\tilde{C}\| \leq 1$. Instead we give an algorithm that performs multiple calls to an SDP-solver, each of which has constant ε but higher r.

Lemma 56. *Assume an SDP-solver that costs*

$$C(n, m, s, R, \varepsilon, r)$$

to solve one SDP with precision ε, and assume that C is non-decreasing in r. Every SDP with parameters $R, r \geq 1$ can be solved up to precision $0 < \varepsilon \leq 1$ with cost

$$\sum_{k=1}^{\log\left(\frac{1}{\varepsilon}\right)} C\left(n+1, m+1, s, R + 4\log\left(\frac{1}{\varepsilon}\right), 1, 2^k(r+1)\right),$$

by solving $\log\left(\frac{1}{\varepsilon}\right)$ SDPs, where the k-th SDP is solved up to precision $\tilde{\varepsilon} = 1$ and has parameters

$$\tilde{n} = n+1,\ \tilde{m} = m+1,\ \tilde{R} = \mathcal{O}\left(R + 4\log\left(\frac{1}{\varepsilon}\right)\right),\ \tilde{r} \leq 2^k(r+1),$$

and input matrices with elements described by bitstrings of length poly$(\log n, \log m, \log\left(\frac{1}{\varepsilon}\right))$. Furthermore, if $C(n, m, s, R, 1, r) =$ poly(n, m, s, R, r), then the above cost is poly$(n, m, s, R, r/eps)$.

Proof. The high-level idea is that we want to learn a small interval in which the optimum lies, whilst using a "big" precision of 1. We do so as follows: given an interval $[L, U]$ with the promise that $\mathrm{OPT} \in [L, U]$, we formulate another SDP for which a 1-approximation of the optimum learns us a new, smaller, interval $[L', U']$ such that $\mathrm{OPT} \in [L', U']$. We will moreover have $U' - L' \leq \frac{1}{2}(U - L)$. In the remainder of the proof we first show how to do this reformulation, we then use this technique to prove the lemma.

Given an SDP p, given in the form of Equation (1), of which we know an interval $[L, U]$ such that $\mathrm{OPT} \in [L, U]$ (with $0 < U - L \leq 1$), we can write down an equivalent SDP p' such that an optimal solution of p corresponds one-to-one to an optimal solution of p', and the optimum

of p' lies in $[0,4]$:

$$
\begin{aligned}
(p') \qquad \max \quad & \mathrm{Tr}\left(\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\tilde{X}\right) \\
\text{s.t.} \quad & \mathrm{Tr}\left(\begin{pmatrix} -C & 0 \\ 0 & \frac{U-L}{4} \end{pmatrix}\tilde{X}\right) \leq -L, \\
& \mathrm{Tr}\left(\begin{pmatrix} A_j & 0 \\ 0 & 0 \end{pmatrix}\tilde{X}\right) \leq b_j \quad \text{for all } j \in [m], \\
& \tilde{X} \succeq 0,
\end{aligned}
$$

here the variable $\tilde{X}$ is of size $(n+1)\times(n+1)$, and should be thought of as $\tilde{X} = \begin{pmatrix} X & \cdot \\ \cdot & z \end{pmatrix}$, where X is the variable of the original SDP: an $n \times n$ positive semidefinite matrix. Observe that by assumption, for every feasible X of the original SDP, $L \leq \mathrm{Tr}(CX) \leq U$. Therefore, the first constraint implies $0 \leq z \leq 4$ and hence the new optimum lies between 0 and 4, and the new trace bound is $\tilde{R} = R + 4$. We now determine $\tilde{r}$. The dual of the above program is given by:

$$
\begin{aligned}
(d') \qquad \min \quad & -Ly_0 + \sum_{j=1}^{m} b_j y_j \\
\text{s.t.} \quad & \begin{pmatrix} -C & 0 \\ 0 & \frac{U-L}{4} \end{pmatrix} y_0 + \sum_{j=1}^{m} \begin{pmatrix} A_j & 0 \\ 0 & 0 \end{pmatrix} y_j \succeq \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \\
& y \geq 0.
\end{aligned}
$$

Claim 57. *An optimal solution $\tilde{y}$ to d' is of the form $\tilde{y} = \frac{4}{U-L}(1, y)$ where y is an optimal solution to d, the dual of p.*

The proof of the claim is deferred to the end of this section. The claim implies that $\tilde{r} = \frac{4}{U-L}(1+r)$. We also have

$$\mathrm{OPT} = L + \mathrm{OPT}' \frac{U-L}{4},$$

and hence, a 1-approximation to OPT' gives a $\frac{U-L}{4}$-approximation to OPT.

We now use the above technique to prove the lemma. Assume an SDP of the form (1) is given. By assumption, $\|C\| \leq 1$ and therefore $\mathrm{OPT} \in [-R, R]$. Calling the SDP-solver on this problem with $\varepsilon = 1$ will give us an estimate of OPT up to additive error 1. Call this estimate OPT_0, then $\mathrm{OPT} \in [\mathrm{OPT}_0 - 1, \mathrm{OPT}_0 + 1] =: [L_0, U_0]$. We now define a new SDP p' as above, with $U = U_0, L = L_0$ (notice $U_0 - L_0 \leq 2$). By the above, solving p' with $\tilde{\varepsilon} = 1$ determines a new interval $[L_1, U_1]$ of length at most $2\frac{U_0-L_0}{4}$ such that $\mathrm{OPT} \in [L_1, U_1]$. We use the interval $[L_1, U_1]$ to build a new SDP p' and solve that with $\tilde{\varepsilon} = 1$ to get an interval $[L_2, U_2]$. Repeating this procedure $k = \log\left(\frac{1}{\varepsilon}\right) + 1$ times determines an interval $[L_k, U_k]$ of length at most $\frac{1}{2^k}(U_0 - L_0) \leq \varepsilon$. Hence, we have determined the optimum of p up to an additive error of ε. The total time needed for this procedure is at most

$$\sum_{k=1}^{\log\left(\frac{1}{\varepsilon}\right)} C\left(n+1, m+1, s, R + 4\log\left(\frac{1}{\varepsilon}\right), 1, 2^k(r+1)\right). \qquad \square$$

Proof of Claim 57. First observe that the linear matrix inequality in d' implies the inequality $y_0\frac{U-L}{4} \geq 1$ and hence $y_0 \geq \frac{4}{U-L}$. Suppose we fix a value y_0 (with $y_0 \geq \frac{4}{U-L}$) in d', then the resulting SDP is of the form

$$(d'') \qquad -Ly_0 + \min \quad \sum_j b_j y_j$$
$$\text{s.t.} \quad \sum_{j=1}^{m} y_j A_j \succeq y_0 C,$$
$$y \geq 0.$$

and hence, an optimal solution $\tilde{y}$ to d'' is of the form $\tilde{y} = y_0 y$ where y is an optimal solution to d. It follows that an optimal solution to d' is of the form $(y_0, y_0 y)$ where y is an optimal solution to d. Observe that the optimal value of d' as a function of y_0 is of the form $y_0 \cdot (-L + \text{OPT})$. Since by assumption $\text{OPT} \geq L$, the objective is increasing (linearly) with y_0 and hence $y_0 = \frac{4}{U-L}$ is optimal. □