58th Annual IEEE Symposium on Foundations of Computer Science

Quantum Speed-ups for Solving Semidefinite Programs

Fernando G.S.L. Brandao
Institute of Quantum Information and Matter
California Institute of Technology
Pasadena, CA
Email: fgslbrandao@gmail.com

Krysta M. Svore
Quantum Architectures and Computation Group
Microsoft Research
Redmond, WA
Email: ksvore@microsoft.com

***Abstract*—We give a quantum algorithm for solving semidefinite programs (SDPs). It has worst-case running time $n^{\frac{1}{2}} m^{\frac{1}{2}} s^2 \operatorname{poly}(\log(n), \log(m), R, r, 1/\delta)$, with n and s the dimension and row-sparsity of the input matrices, respectively, m the number of constraints, δ the accuracy of the solution, and R, r upper bounds on the size of the optimal primal and dual solutions, respectively. This gives a square-root unconditional speed-up over any classical method for solving SDPs both in n and m. We prove the algorithm cannot be substantially improved (in terms of n and m) giving a $\Omega(n^{\frac{1}{2}} + m^{\frac{1}{2}})$ quantum lower bound for solving semidefinite programs with constant s, R, r and δ.**

The quantum algorithm is constructed by a combination of quantum Gibbs sampling and the multiplicative weight method. In particular it is based on a classical algorithm of Arora and Kale for approximately solving SDPs. We present a modification of their algorithm to eliminate the need for solving an inner linear program which may be of independent interest.

***Keywords*-quantum algorithms; semidefinite programs; Gibbs sampling**

I. Introduction

Quantum computers harness the unique features of quantum mechanics to compute in novel ways and outperform classical solutions to some problems. In the past 25 years a variety of quantum algorithms offering speed-ups over classical computations have been found, including Shor's polynomial-time quantum algorithm for factoring [1] and Grover's quantum algorithm for searching a database in time square-root its size [2]. A central challenge in quantum computing is to identify more quantum algorithms that outperform their classical computing counterparts, especially for practically relevant problems.

Semidefinite programming is one of the most successful algorithmic frameworks of the past few decades [3]. It has applications ranging from designing efficient algorithms for approximating combinatorial optimization problems [4] to operations research and beyond [5]. The power of semidefinite programs (SDPs) resides in their generality, together with the fact that there are efficient methods for solving them [5]. However, given the steadily increasing sizes of SDPs found in practice, it is an important problem to find even more efficient algorithms.

We present a quantum algorithm for solving semidefinite programs achieving a quadratic speed-up over any classical method, both in the dimension of the matrices and in the number of constraints of the program (we note however that the algorithm run-time also depends on a parameter measuring the size of the solution of the SDP, discussed below). Our work also shows the first quantum speed-up for linear programming, which is an important subclass of SDPs. We show that such a quadratic speed-up is not far from the best possible in general. We believe the results herein make a compelling case that solving SDPs efficiently has the potential to be a relevant application of future quantum computers.

A. Semidefinite Programs

A general semidefinite program (SDP) is given by

$$\begin{aligned} & \max \operatorname{tr}(CX) \\ \forall j \in [m], \quad & \operatorname{tr}(A_j X) \leq b_j \\ & X \geq 0, \end{aligned} \tag{1}$$

where the $n \times n$ Hermitian matrices $(C, A_1, \ldots, A_m)$ and the real numbers $(b_1, \ldots, b_m)$ are the inputs of the problem. The optimization is taken over positive semidefinite $n \times n$ matrices X. The dual program of the primal SDP given by Eq. (1) is the following:

$$\begin{aligned} & \min b.y \\ & \sum_{j=1}^{m} y_j A_j \geq C \\ & y \geq 0, \end{aligned} \tag{2}$$

where the minimization is taken over real vectors $y := (y_1, \ldots, y_m)$. Under mild conditions the primal and dual problems have the same optimal value [5]. We can assume that

$$\|A_i\| \leq 1 \ \ \forall i \in [m], \ \text{ and } \ \|C\| \leq 1, \qquad (3)$$

with $\| * \|$ the operator norm. This is without loss of generality by normalizing b_i and the optimal solution appropriately.

There are several different classical polynomial-time algorithms for solving semidefinite programs. One is the class of interior point methods. The state-of-the-art algorithm (in terms of rigorous worst-case bounds) has running time $\tilde{O}(m(m^2 + n^{\omega} + mns)\log(1/\delta))$ [7], with ω the exponent of matrix multiplication, s the row-sparsity of the matrices $(C, A_1, \ldots, A_m)$ (i.e., the maximum number of non-zero entries in each of the rows of the matrices), and δ the accuracy of the solution.

If one is willing to tolerate a worse scaling with error, faster algorithms can be sometimes obtained using the multiplicative weight method [8], [9], [10]. In particular Arora and Kale gave an algorithm for solving the SDP given by Eq. (1) in time $\tilde{O}(nms\left(\frac{Rr}{\delta}\right)^4 + ns\left(\frac{Rr}{\delta}\right)^7)$ [9], where R and r are upper bounds on the size of the optimal primal and dual solutions, respectively.

It is an important open problem to find even more efficient algorithms for solving SDPs. Nonetheless, as we show in Section II-C, a limit for any such improvement is the lower bound of $\Omega(n+m)$ for SDPs with $s, \omega, R = O(1)$.

B. Problem Statement

The problem we want to solve is the following: Given a list of $n \times n$ matrices $(A_1, \ldots, A_m, A_{m+1})$ (with $A_{m+1} := C$), approximate the optimal value of Eqs. (1) and (2) and output optimal primal and/or dual solutions. To formulate the problem precisely we must specify in which form the inputs and outputs are given.

Input Model: We assume there is an oracle $\mathcal{P}_A$ that given the indices $j \in [m+1]$, $k \in [n]$ and $l \in [s]$, computes a bit string representation of the l'th non-zero element of the k-th row of A_j,[1] i.e., the oracle performs the following map:

$$|j, k, l, z\rangle \to \left|j, k, l, z \oplus (A_j)_{k f_{jk}(l)}\right\rangle \qquad (4)$$

with $f_{jk} : [r] \to [N]$ a function (parametrized by the matrix index j and the row index k) which given $l \in [s]$ computes the column index of the l-th non-zero entry.

Output: One way to specify the output is to require a list with the entries of X or y. However it is clear that in such a case at least $n(n-1)/2$ (or m) time is required even to write down a primal (or dual) solution. Therefore it is necessary to relax the format of the output in order to obtain faster algorithms.

We require the quantum algorithm provides the following:

- An estimate of the optimal objective value.
- An estimate of $\|y\|_1$ and/or $\operatorname{tr}(X)$.
- *Samples* from the distribution $p := y/\|y\|_1$ and/or from the quantum state $\rho := X/\operatorname{tr}(X)$.[2]

As we show in Section II-C, $\Omega(n+m)$ calls to the oracle are required even classically to output a solution as above.

II. Results

In this paper we give the first quantum algorithm for solving SDPs offering a speed-up over classical methods. Below we state the main contributions on a high level, describe the algorithm, and present a few open questions related to it.

A. Main Ideas

The first contribution of the paper is to notice that classical algorithms for solving SDPs based on the multiplicative weight method [9] imply that in order to solve the SDP of Eq. (1), it is enough to prepare quantum thermal (Gibbs) states of Hamiltonians given by linear combinations of the input matrices $A_1, \ldots, A_m, C$ of the program. Therefore in cases where such Gibbs states can be prepared efficiently (in time polynomial in $\log(n)$), quantum computers can give exponential speed-ups. This already suggests that our method might be an interesting heuristic to run on quantum computers (e.g., by using quantum Metropolis sampling [11], [12] to prepare the Gibbs states).

The second contribution is to combine the first observation with amplitude amplification [17] in the preparation of the Gibbs state to achieve a generic quadratic speed-up in terms of n, the dimension of the input matrices of the program. Here we can apply known results [13], [14] on using amplitude amplification to prepare Gibbs states on a quantum computer in time given roughly by the square root of the dimension of the system.

[1]We assume all elements of the input matrices can be represented exactly by a bit string of size $\operatorname{polylog}(n, m)$. If not we can truncate the matrices, which will only incur error $\exp(-\operatorname{polylog}(n, m))$ that can be neglected.

[2]In this paper we present a quantum algorithm producing an estimate of $\|y\|_1$ and samples from the probability distribution $p := y/\|y\|_1$. The algorithm can be modified to also generate an estimate of $\operatorname{tr}(X)$ and samples from $\rho := X/\operatorname{tr}(X)$. However, we leave the details of this improvement to a future version of the paper.

The third contribution is to show one can achieve a quadratic speed-up also in m, the number of constraints of the program. Establishing this fact requires more work. We modify the Arora-Kale algorithm and replace the inner linear program they use by the preparation of a Gibbs state of a classical Hamiltonian, whose entries are estimated from the expectation value (with each of the input matrices) of the Gibbs state prepared in the main thread of the algorithm. We show this replacement is possible using Jaynes' principle of maximum entropy [18], or more specifically a recent approximate version of it [19] with better control of parameters. Then we apply amplitude amplification also to the preparation of this Gibbs state. This modification alone is not enough to give a speed-up, since the sparsity of the Hamiltonians for which we must prepare the associated Gibbs states also depends on m (and the quantum algorithms for preparing Gibbs states we consider have linear dependence on sparsity). We then show that we can sparsify the Hamiltonians under consideration by random sampling, without changing the functioning of the algorithm, so that their sparsity only depends on the original sparsity s of the input matrices (up to polylogarithmic factors in n, m).

B. The Algorithm

Reduction to Feasibility: Using binary search we can reduce the optimization problem to a feasibility one. Let α be a guess for the optimal solution (which will be varied by binary search). We are then concerned with the problem of either sampling from a probability distribution $p := y/\|y\|_1$, with y a dual feasible vector whose value is at most $\alpha(1+\delta)$ for a small $\delta > 0$, or finding out that the optimal value is greater than $\alpha(1-\delta)$.[3]

Gibbs Samplers: A subroutine of the main algorithm is the following:

Definition 1 (Gibbs Sampler). *Let H be a Hamiltonian and $O[H]$ an oracle for its entries.*[4] *Then* $\mathsf{GibbsSampler}(O[H], \nu)$ *is a quantum operation that given access to $O[H]$, outputs a state ρ such that $\|\rho - e^H/\operatorname{tr}(e^H)\|_1 \leq \nu$.*

Several different Gibbs samplers have been proposed in the literature [11], [12], [13], [14], [15], [16], and any of them could be used in the main quantum algorithm.

Oracle by Estimation: Given a Hamiltonian H, in order to run a Gibbs sampler algorithm, we need an oracle for its entries. We also need the notion of a probabilistic oracle. This is an oracle that with high probability outputs the right entry of the Hamiltonian, but with small probability might output a wrong value.

Consider a quantum state ρ and two real numbers λ and μ. We define the Hamiltonian $h(\rho, \lambda, \mu) := \sum_{i=1}^m r_i |i\rangle\langle i|$, with

$$r_i := \lambda \operatorname{tr}(A_i \rho) + \mu b_i, \tag{5}$$

and its truncated version

$$\overline{h}(\rho, \lambda, \mu) := \sum_{i=1}^m \overline{r}_i |i\rangle\langle i|, \tag{6}$$

with $\overline{r}_i$ the rounding of r_i to precision $h_{\text{precision}}$. Throughout the paper we set

$$h_{\text{precision}} = \frac{\delta}{56R^2}. \tag{7}$$

The quantum algorithm will make use of calls to a probabilistic oracle for $\overline{h}$, which we show how to construct in Section IV. A subtlety is that ρ will not be given explicitly as a density matrix, but only as a quantum state. Therefore in order to implement the oracle for $\overline{h}$, we need to first estimate (some of) the values $\{\operatorname{tr}(A_i\rho)\}_{i=1}^m$.

The Size Parameter R: Apart from the dimension of the matrices n, the number of constraints m, the sparsity of the input matrices s, and the error δ, the algorithm will depend on another parameter of the SDP. For many problems of interest, this is a constant independent of n and m.[5]

Following [9], we assume $A_1 = I$ and let $b_1 = R$. Thus we have the constraint

$$\operatorname{tr}(X) \leq R. \tag{8}$$

We can always add this constraint without changing the optimal solution by choosing R sufficiently large. The parameter R is a measure of the size of the optimal primal solution of the SDP. Note we have the upper bound $\alpha \leq R$.[6] We also assume R is chosen sufficiently large such that $\max_i |b_i| = R$.

Reduction to $b_i \geq 1$ and $\alpha \geq 1$ and the dual size parameter r: Our algorithm will only work for SDPs

[3]Alternatively we could also sample from a quantum state $\rho := X/\operatorname{tr}(X)$, with X a primal feasible solution with objective value at least $\alpha(1-\delta)$; however we do not consider this task in the current version.

[4]In analogy with the input model 1, $O[H]$ is an oracle that given the indices $k \in [n]$ and $l \in [s]$, computes a bit string representation of the l'th non-zero element of the k-th row of H. Here n is the dimension of H and s its sparsity.

[5]We remind the reader we are also assuming that $\|C\|, \|A_i\| \leq 1$ for all $i \in [m]$ (which is w.l.o.g. by changing the values of the numbers b_j, α appropriately).

[6]It follows from $\operatorname{tr}(CX) \leq \|X\|_1 \|C\| \leq \operatorname{tr}(X) \leq R$.

for which $b_i \geq 1$ for all $i \in [m]$. However in the full version of the paper, we prove Lemma 2 below, which shows that if we can solve SDPs with the b_i's greater than one, we can solve arbitrary SDPs in roughly the same time. We say a feasible solution is δ-optimal if its objective value is within additive error δ to the optimal. Given the SDP of Eq. (2) with an optimal solution $(y_1, \ldots, y_m)$, we assume we are given an upper bound r to the sum $\sum_{i=1}^m y_i$ (if there is more than one solution, any of them can be chosen). We have:

Lemma 2. *One can sample from a δ-optimal solution of the SDP given by Eq. (2) (with dimension n, m variables, size parameter R and upper bound r on optimal solution vector) given the ability to sample from a (δ/r)-optimal solution of the SDP given by Eq. (2) (with dimension $n+1$, $m+1$ variables and size parameter $2R+1$) in which $b_i \geq 1$ for all $i \in [m]$.*

To apply Lemma 2 we need an upper bound r on the ℓ_1-norm of the optimal dual solution vector. For example, if all b_i's are positive, we can take $r = \alpha / \min_i b_i$.

We also assume $\alpha \geq 1$. We ensure this is the case as follows: Given the SDP of Eq. (2) with all b_i's greater than one, it is clear that we can take $\alpha > 0$ (as all the y_i's are non-negative). Suppose $\alpha < 1$. Then we consider a new SDP with the b_i's rescaled by $1/\alpha$. As an effect we must solve the scaled SDP to accuracy $\delta\alpha$. Note the complexity of solving the SDP (which depends on the accuracy) increases as α approaches zero. It is an open question if this drawback can be avoided.

The quantum algorithm for $\alpha \geq 1$, $b_i \geq 1$ discussed below is given in terms of multiplicative error, while the reduction above works with additive error. It is easy to convert the multiplicative approximation to additive approximation by redefining the error $\delta \to \delta/\alpha$. This results in an increased complexity of the algorithm in terms of α.

Probability of success: The probability of success of our algorithm is greater than

$$1 - O(\exp(-log^{\xi}(nm))), \tag{9}$$

where $\xi > 0$ is a free parameter.

Algorithm 3 (Quantum Algorithm for Solving SDPs)**.**

```
Input: Oracles for ${A_1, ..., A_m, C}$, with $\|C\|, \|A_i\| \leq 1$ and ${b_1, ..., b_m}$ with $b_i \geq 1$.
       Parameters $R, \alpha, \delta, \xi > 0$.
Output: Either a sample from distribution $p$ and a real number $L$ such that $y := Lp$ is dual feasible
        with objective value less than $(1+\delta)\alpha$, or the label Greater indicating the optimal
        objective value is greater than $(1-\delta)\alpha$.

Set $\rho^{(1)} = I/n$. Let $\varepsilon = \frac{\delta}{28R^2}$, $\varepsilon' = -\ln(1-\varepsilon)$, $M = 80\log^{1+\xi}(8R^2nm/\varepsilon)/\varepsilon^2$, $L = 80\log^{1+\xi}(nm)/\varepsilon^2$
and $Q = 10^6 R^6 \ln^{2+\xi}(nm)/\delta^4$. Let $T = \frac{500R^3\ln(n)}{\delta^2}$.
For $t = 1, ..., T$:
  1) Set $y^{(t)} = (0, ..., 0)$.
  2) For $k = 1, ..., \gamma = \lceil \frac{8}{\varepsilon^2}\log(m)R^2 \rceil$,
        $N = 1, ..., \lceil \frac{\alpha}{\varepsilon} \rceil$,
       • Create $M$ copies of
         $q \leftarrow$ GibbsSampler$(O[\overline{h}(\rho^{(t)}, k)], \varepsilon/4)$.
       • Sample $i_1, ..., i_M$ independently from the distribution $q$.
       • Compute estimates ${e_{i_1}, ..., e_{i_M}, f}$ of
         ${\mathrm{tr}(A_{i_1}\rho^{(t)}), ..., \mathrm{tr}(A_{i_M}\rho^{(t)}), \mathrm{tr}(C\rho^{(t)})}$
         to accuracy $\varepsilon/2$ using $(M+1)L$ samples from $\rho^{(t)}$ .
       • If $1/M\sum_{j=1}^M e_{i_j} \geq f/(\varepsilon N) - \varepsilon$ and
         $1/M\sum_{j=1}^M b_{i_j} \leq \alpha/(\varepsilon N) + R\varepsilon$, set $k_t = k$,
         $N_t = N$, $q^{(t)} = q$ and $y^{(t)} = \varepsilon N q^{(t)}$.
  3) If $y^{(t)} = (0, ..., 0)$, stop and output Greater.
  4) Create $Q+1$ copies of
     $q^{(t)} \leftarrow$ GibbsSampler$(O[\overline{h}(\rho^{(t)}, k_t)], \varepsilon/4)$
  5) Sample $i_1, ..., i_Q$ independently from the distribution $q^{(t)}$.
  6) Let
     $M^{(t)} = \left(\varepsilon N_t Q^{-1}\sum_{j=1}^{Q} A_{i_j} - C + 2\alpha I\right)/4\alpha.$
  7) Let $C_t := \frac{10\log(m)}{\varepsilon^2}(\frac{\gamma\alpha}{\varepsilon}M + Q)G_{\overline{h}}(\rho^{(t)}) + \frac{2\gamma\alpha}{\varepsilon}ML$.
     Create $C_t$ copies of the state
     $\rho^{(t+1)} \leftarrow$ GibbsSampler$(-\varepsilon'(\sum_{\tau=1}^{t} M^{(\tau)}), \varepsilon/4)$.
Output: $\|\overline{y}\|_1$ and a sample from $\overline{y}/\|\overline{y}\|_1$ with
        $\overline{y} = \frac{\delta\alpha}{2R}e_1 + \frac{1}{T}\sum_{t=1}^{T} y^{(t)}$.
```

The Algorithm: Let

$$\gamma := \left\lceil \frac{8}{\varepsilon^2}\log(m)R^2 \right\rceil \tag{10}$$

for an $\varepsilon > 0$ defined below. For an integer $k \leq \gamma$, we define

$$\overline{h}(\rho, k) := \overline{h}\left(\rho, -\frac{\varepsilon}{8R^2}k, -\frac{\varepsilon}{8R^2}(\gamma - k)\right), \tag{11}$$

with $\overline{h}$ the Hamiltonian given by Eq. (6). Let $e_1 := (1, 0, \ldots, 0)$ be the first computational basis state of $\mathbb{R}^m$. Let

$$G_{\overline{h}}(\rho) := \max_{k \in [\gamma]} \Bigg(\text{number of calls to the oracle } O[\overline{h}(\rho, k)] \text{ in } \mathsf{GibbsSampler}\left(O[\overline{h}(\rho, k)], \frac{\varepsilon}{4}\right)\Bigg). \tag{12}$$

The main algorithm is given in Algorithm 3.

Let

$$G_M := \max_{t \leq T} \Bigg(\text{number of calls to the oracle in } \mathsf{GibbsSampler}\left(O\left[-\varepsilon'\left(\sum_{\tau=1}^{t} M^{(\tau)}\right)\right], \frac{\varepsilon}{4}\right)\Bigg), \tag{13}$$

and

$$G_{\overline{h}} := \max_{t \leq T} G_{\overline{h}}(\rho^{(t)}). \tag{14}$$

Finally, let T_{Meas} be the maximum time needed to estimate one of $\operatorname{tr}(A_i\rho)$, for $i \in [m]$, and $\operatorname{tr}(C\rho)$ (for an arbitrary ρ) within additive error $\varepsilon/2$ (in Lemma 12 we show that for s-sparse matrices, $T_{\text{Meas}} \leq \tilde{O}(s/\varepsilon^2)$).

We prove in Section V the following:

Theorem 4. *Algorithm 3 runs in time*

$$\tilde{O}\left(\frac{R^{21}}{\delta^{11}} G_{\overline{h}} G_M\right) + \tilde{O}\left(\frac{R^{13}}{\delta^5} T_{Meas}\right). \tag{15}$$

The algorithm fails with probability at most

$$O\left((R/\delta)^{18}(nm)^{10}\exp\left(-\log^{\xi}(nm)\right)\right). \tag{16}$$

Assuming it does not fail, if it outputs Greater*, the optimal objective value is greater than $(1-\delta)\alpha$. Otherwise it outputs a sample of a probability distribution p and a real number L such that $y = Lp$ is dual feasible and $\sum_i y_i b_i \leq (1+\delta)\alpha$.*

We note the algorithm is very costly in terms of the size parameter R and the error δ. We believe it is possible to significantly reduce the complexity in terms of these two parameters, but we leave this possibility as an open question to future work.

One interesting Gibbs sampler to consider is quantum Metropolis [11], [12]. Although it is difficult to obtain rigorous estimates on its running time, it is expected that it is polylogarithmic in many cases. Whenever this is the case for the Hamiltonians involved in the algorithm (given by linear combinations of the A_i's and C), one would achieve exponential speed-ups.

In Section V we show:

Corollary 5. *Using the Gibbs Sampler from Ref. [13], Algorithm 3 runs in time $\tilde{O}(n^{\frac{1}{2}} m^{\frac{1}{2}} s^2 R^{32}/\delta^{18})$.*

As we show in the next section, this represents an unconditional polynomial speed-up (in terms of m and n) over any classical method for solving semidefinite programming.

One particular case of interest is when R is a constant independent of all other parameters. In this case the running time only depends on the parameters n, m, s, δ. Moreover the SDP has a clear quantum interpretation: we want to optimize the expectation value of an observable C/R on a quantum state ρ subject to the constraints that the expectation value of A_i on ρ is bounded by b_i/R, for all $i \in [m]$.

C. Lower Bounds

We give a lower bound on the complexity of solving SDPs which shows the n, m dependence of Algorithm 3 cannot be substantially improved. Consider the following two instances of the primal problem given by Eq. (1), with $R = 1$, $A_1 = I$, $b_j = 1$ for $j \in [m]$ and either

1) For a random $i \in [n]$, set $C_{ii} = 1$. All other elements of C are set to zero. Choose at random $j \in [m]$ and set $(A_j)_{ii} = 2$. All other elements of the matrices $\{A_j\}_{i=2}^m$ are set to zero.
2) For a random $i \in [n]$, set $C_{ii} = 1$. All other elements of C are set to zero. All elements of the matrices $\{A_j\}_{i=2}^m$ are set to zero.

We claim that to decide which of the two cases we are given requires at least $\Omega(n+m)$ calls to the oracle classically and $\Omega(\sqrt{n} + \sqrt{m})$ calls quantum-mechanically. This follows from an elementary reduction to the search problem.

It is easy to see that the optimal solution of the primal and dual problems in the first case are $X = |i\rangle\langle i|/2$ and $y = |j\rangle\langle j|$, with objective value $1/2$. In the second case, in turn, $X = |i\rangle\langle i|$ and $y = |1\rangle\langle 1|$, with objective value 1. Therefore we can decide which of the two we are given and find the marked (i, j) (in the first case) given samples of the optimal y and X and a constant-error approximation to $\operatorname{tr}(X)$ or $\|y\|_1$. This is equivalent to solving two search problems, one in a list of n elements and another in a list of m elements.

D. Discussion and Open Questions

The core quantum part of the algorithm is the preparation of quantum Gibbs states. Classically there are several interesting applications of the Monte Carlo method and the Metropolis algorithm to problems not related to simulating thermal properties of physical systems [24]. One could expect the same will be the case for quantum Metropolis. We might have to wait until there are working quantum computers to fully explore the usefulness of quantum Metropolis, since in analogy to the classical case, many times heuristic methods based on it might work well in practice even though it is hard to get theoretical guarantees. Nevertheless, as far as we know the results of this paper give the first example of a problem of interest outside the simulation of physical systems in which ”quantum Monte Carlo” methods (i.e., sampling from quantum Gibbs states) play an important role. The algorithm can also be seen as a new application of quantum annealing. One difference is that in this case the annealing is used to prepare a finite temperature state, instead of a groundstate as is usually considered in quantum adiabatic optimization.

The algorithm is also inherently robust, in the sense that to compute a solution of the SDP to accuracy ε, it suffices to be able to prepare approximations to accuracy $O(\varepsilon)$ of the Gibbs state of Hamiltonians given by linear combinations of the input matrices. Moreover we believe the constants and the dependence on R and δ might be substantially improved by a more careful analysis. If this turns out to be indeed the case, we expect the algorithm to be a promising candidate for a relevant application of small quantum computers (even without the need for error correction).

This work leaves several open questions for future work. For example:

- The algorithm has very poor scaling in terms of R and δ. It is a pressing open question to improve its running time in terms of these parameters. Also can we close the gap in terms of n and m between the lower bound ($\Omega(\sqrt{n}+\sqrt{m})$) and the algorithm ($O(\sqrt{nm})$)?
- Although quadratic speed-ups in terms of n and m are the best possible in the worst case, it is an interesting question whether more significant speed-ups are possible in specific instances. How large are the speed-ups on average (for example choosing the input matrices at random from a given distribution)? Even more interesting is to explore whether there is an SDP of practical interest for which we might have greater quantum speed-ups.
- How robust is the algorithm to noise? Can we run it without the need of quantum error correction in analogy to what has been proposed for quantum annealing? Is there an improvement of the algorithm which would be suitable for a small-scale quantum computer (with hundreds of physical qubits)?
- The multiplicative weight method is an important algorithmic technique classically. In this paper we give an application of the matrix multiplicative weight method to quantum algorithms. Are there more applications?
- Can we enlarge the class of optimization problems having a quantum speed-up beyond SDPs? In particular, can we get quantum speed-ups for optimizing general convex functions over convex sets (assuming we have an efficient oracle for membership in the set)?
- In practice the preferred algorithms for solving SDPs are based on the interior point method. Can we also find a quantum algorithm for SDPs based on it?

III. Analysis of the Quantum Algorithm

A. The Arora-Kale Algorithm

The quantum algorithm builds on a classical algorithm of Arora and Kale for solving SDPs, which we now review. One element of their approach is an auxiliary algorithm termed $\mathsf{ORACLE}(\rho)$, which given a density matrix ρ, searches for a vector y from the polytope

$$\mathcal{D}_\alpha := \{y \in \mathbb{R}^m : y \geq 0, b.y \leq \alpha\} \tag{17}$$

such that

$$\sum_{j=1}^{m} y_j \operatorname{tr}(A_j\rho) \geq \operatorname{tr}(C\rho), \tag{18}$$

or outputs *fail* if no such vector exists.

The running time of their algorithm also depends on the so-called width ω of the SDP, defined as

$$\omega := \max_{y\in\mathcal{D}_\alpha} \left\| \sum_j y_j A_j - C \right\|. \tag{19}$$

We note the bound:[7]

$$\begin{aligned}\omega &= \max_{y\in\mathcal{D}_\alpha}\left\|\sum_j y_j A_j - C\right\| \\ &\leq \max_{y\in\mathcal{D}_\alpha}\sum_j y_j + 1 \\ &\leq \max_{y\in\mathcal{D}_\alpha}\sum_j b_j y_j + 1 \\ &\leq \alpha + 1 \\ &\leq R + 1. \end{aligned} \tag{20}$$

The Arora-Kale algorithm is the following:

Algorithm 6 (Arora-Kale Algorithm for SDPs)**.**

Set $\rho^{(1)} = I/n$. Let $\varepsilon = \frac{\delta\alpha}{2R^2}$, and let $\varepsilon' = \ln(1-\varepsilon)$. Let $T \geq \frac{16R^4\ln(n)}{\alpha^2\delta^2}$. For $t = 1, ..., T$:

```
1) Run ORACLE($\rho^{(t)}$). If it fails, stop and output $\rho^{(t)}$.
2) Else, let $y^{(t)}$ be the vector generated by ORACLE($\rho^{(t)}$).
3) Let $M^{(t)} = (\sum_{j=1}^m A_j y_j^{(t)} - C + \omega I)/2\omega$
4) Compute $W^{(t+1)} = \exp(-\varepsilon'(\sum_{\tau=1}^t M^{(\tau)}))$.
5) Set $\rho^{(t+1)} = \frac{W^{(t+1)}}{\mathrm{tr}(W^{(t+1)})}$ and continue.
```

The central idea behind the algorithm is a variant of the multiplicative weight method for positive semidefinite matrices. Let us denote by $\lambda_n(X)$ the minimum eigenvalue of the $n \times n$ Hermitian matrix X. Arora and Kale proved the following:

Lemma 7. *[Matrix Multiplicative Weights method; Theorem 10 of [9]] Let $M^{(t)}$ be such that $0 \leq M^{(t)} \leq I$ for every t. Fix $\varepsilon < \frac{1}{2}$, and let $\varepsilon' = -\ln(1-\varepsilon)$. define $W^{(t)} = \exp\left(-\varepsilon'\left(\sum_{\tau=1}^{t-1} M^{(\tau)}\right)\right)$ and the density matrices $\rho^{(t)} = \frac{W^{(t)}}{\mathrm{tr}(W^{(t)})}$. Then*

$$\sum_{t=1}^T \mathrm{tr}(M^{(t)}\rho^{(t)}) \leq (1+\varepsilon)\lambda_n\left(\sum_{t=1}^T M^{(t)}\right) + \frac{\ln(n)}{\varepsilon}. \tag{21}$$

Let $e_1 := (1, 0, \dots, 0)$. Then the main result of [9] is the following (as a warm up to the proof of correctness of the quantum algorithm we reproduce the argument of Arora and Kale below):

Theorem 8 (Theorem 1 of [9])**.** *Suppose* ORACLE *never fails for $T = \frac{16R^4\ln(n)}{\alpha^2\delta^2}$ iterations. Then $\overline{y} = \frac{\delta\alpha}{R}e_1 + \frac{1}{T}\sum_{t=1}^T y^{(t)}$ is dual feasible with objective value at most $\alpha(1+\delta)$.*

[7]Assuming $b_i \geq 1$.

The proof is given in [9] and in the full version of the paper.

B. Approximately Implementing ORACLE *by Gibbs Sampling*

As a step towards the quantum algorithm, we now give an explicit implementation of the ORACLE auxiliary algorithm. The idea is to use the fact that by Jaynes' principle, we can w.l.o.g. take the output y of ORACLE(ρ) to be (up to normalization) a Gibbs probability distribution over the two constraints, i.e., a distribution over $[m]$ of the form

$$q_{\rho,\lambda,\nu}(i) := \frac{\exp\left(\lambda\,\mathrm{tr}(A_i\rho) + \mu\sum_i b_i\right)}{\sum_i \exp\left(\lambda\,\mathrm{tr}(A_i\rho) + \mu\sum_i b_i\right)}, \tag{22}$$

for real numbers λ, μ.

The following is a special case of Lemma 4.6 of [19] (obtained by taking the reference state to be maximally mixed) and is an approximate version of Jaynes' principle with a quantitative control of the parameters of the Hamiltonian in the Gibbs state (in contrast, in the original Jaynes' principle there is no control over the size of the interaction strengths of the Hamiltonian)

Let $\mathcal{M}(\mathbb{C}^n)$ and $\mathcal{D}(\mathbb{C}^n)$ be the set of Hermitian and density matrices over $\mathbb{C}^n$. Let $\mathcal{T} \subseteq \mathcal{M}(\mathbb{C}^n)$ be a compact set of matrices. We set

$$\Delta(\mathcal{T}) := \sup_{A\in\mathcal{T}} \|A\|. \tag{23}$$

For $A \in \mathcal{M}(\mathbb{C}^n)$, we define the associated dual norm

$$[A]_{\mathcal{T}} := \sup_{B\in\mathcal{T}} \mathrm{tr}(BA). \tag{24}$$

Lemma 9. *(Lemma 4.6 of [19]) For every $\kappa > 0$, the following holds. Let $\mathcal{T} \subseteq \mathcal{M}(\mathbb{C}^m)$ be a compact set of matrices and let $\pi \in \mathcal{D}(\mathbb{C}^m)$ be a density matrix. If one defines $\gamma = \lceil \frac{8}{\kappa^2}\log(m)\Delta(\mathcal{T})^2 \rceil$ then there exist $X_1, \dots, X_\gamma \in \mathcal{T}$ such that*

$$\tilde{\pi} := \frac{\exp\left(-\frac{\kappa}{4\Delta(\mathcal{T})^2}\sum_{i=1}^\gamma X_i\right)}{\mathrm{tr}\left(\exp\left(-\frac{\kappa}{4\Delta(\mathcal{T})^2}\sum_{i=1}^\gamma X_i\right)\right)} \tag{25}$$

satisfies

$$[\pi - \tilde{\pi}]_{\mathcal{T}} \leq \kappa. \tag{26}$$

The finitary version of Jaynes' principle above allows us to implement ORACLE assuming we have access to samples from the distributions $q_{\rho,\lambda,\mu}$. In fact, for the quantum algorithm it will be useful to prove a generalization in which we only assume we can sample from distributions $\overline{q}_{\rho,\lambda,\mu}$ *close* in variational distance to $q_{\rho,\lambda,\mu}$.

For an integer k, let

$$q_{\rho,k} := q_{\rho,-\frac{\kappa}{4R^2}k,-\frac{\kappa}{4R^2}(\gamma-k)}, \tag{27}$$

with

$$\gamma := \left\lceil \frac{8}{\kappa^2}\log(m)R^2 \right\rceil. \tag{28}$$

Algorithm 10 (Instantiation of $\mathsf{ORACLE}(\rho)$ by Sampling)**.**

Input: *Samples from distributions $\overline{q}_{\rho,k}$ such that $\|\overline{q}_{\rho,k} - q_{\rho,k}\|_1 \le \nu$ and real numbers $\{e_i\}_{i=1}^{m+1}$ such that $|e_i - \mathrm{tr}(A_i\rho)| \le \nu$. A parameter $\kappa > 0$.*

Output*Samples from distribution $y/\|y\|_1$ and value of $\|y\|_1$ satisfying Eqs. (29) and (30).*

Let $M = 80\log^{1+\xi}(8R^2nm/\varepsilon)/\varepsilon^2$. For $k = 1,\dots,\gamma$ and $N = 1,\dots,\lceil\frac{\alpha}{\kappa}\rceil$:

- *Sample $i_1,\dots,i_M \in [m]^M$ independently from the distribution $\overline{q}_{\rho,k}$.*
- *If $\frac{1}{M}\sum_{j=1}^M e_{i_j} \ge \frac{e_{m+1}}{\kappa N} - (\kappa+\nu)$ and $\frac{1}{M}\sum_{j=1}^M b_{i_j} \le \frac{\alpha}{\kappa N} + R(\kappa+\nu)$, output samples from $\tilde{q}_{\rho,k}$ and the number κN (as $y/\|y\|_1$ and $\|y\|_1$, respectively).*

Lemma 11. *Suppose $\mathsf{ORACLE}(\rho)$ does not fail. Then with probability greater than $1-\exp(-\log^{\xi}(nm))$, Algorithm 10 outputs y such that*

$$\sum_{i=1}^m b_i y_i \le \alpha(1+2R(\kappa+\nu)), \tag{29}$$

and

$$\sum_{i=1}^m y_i\,\mathrm{tr}(A_i\rho) \ge \mathrm{tr}(C\rho) - 2\alpha(\kappa+\nu). \tag{30}$$

Proof:

By the Chernoff bound and the union bound over all all $k \in [\gamma]$, with probability at least $1-\exp(-\log^{\xi}(nm))$, sampling $i_1,\dots,i_M$ independently from $\overline{q}_{\rho,k}$ guarantees that

$$\left|\sum_{i=1}^m \overline{q}_{\rho,k}(i)\,\mathrm{tr}(A_i\rho) - \frac{1}{M}\sum_{j=1}^M e_{i_j}\right| \le \kappa+\nu, \tag{31}$$

and

$$\left|\sum_{i=1}^m \overline{q}_{\rho,k}(i)b_i - \frac{1}{M}\sum_{j=1}^M b_{i_j}\right| \le \kappa+\nu, \tag{32}$$

for all $k \le \gamma$.

Let $k \le \gamma, N \le \lceil\frac{R}{\kappa}\rceil$ be the smallest integers (assuming they exist) such that

$$\sum_{i=1}^m \overline{q}_{\rho,k}(i)\,\mathrm{tr}(A_i\rho) \ge \frac{\mathrm{tr}(C\rho)}{\kappa(N+1)} - (\kappa+\nu), \tag{33}$$

and

$$\sum_{i=1}^m \overline{q}_{\rho,k}(i)b_i \le \frac{\alpha}{\kappa N} + R(\kappa+\nu). \tag{34}$$

Then by Eqs. (31) and (32), with probability greater than $1-\exp(-\log^{\xi}(nm))$ over the choice of $i_1,\dots,i_M$,

$$\frac{1}{M}\sum_{j=1}^M e_{i_j} \ge \frac{\mathrm{tr}(C\rho)}{\kappa(N+1)} - 2(\kappa+\nu), \tag{35}$$

and

$$\frac{1}{M}\sum_{j=1}^M b_{i_j} \le \frac{\alpha}{\kappa N} + 2R(\kappa+\nu), \tag{36}$$

where we used $\max_i |b_i| = R$. The algorithm will then output $y = \kappa N\tilde{q}_{\rho,k}$, which satisfies Eqs. (29) and (30).

It remains to prove the existence of at least one pair $k \le \gamma, N \le \lceil\frac{\alpha}{\kappa}\rceil$ satisfying Eqs. (33) and (34). Let y^* be an output of $\mathsf{ORACLE}(\rho)$. Let us apply Lemma 9 with $\pi := \sum_i y_i^* |i\rangle\langle i|/\|y^*\|_1$ and $\mathcal{T} = \{X := \sum_i \mathrm{tr}(A_i\rho)|i\rangle\langle i|, Y := \sum_i b_i|i\rangle\langle i|\}$. Note that $\Delta(\mathcal{T}) = R$. We find there is an integer $k \le \gamma$ such that

$$\tilde{\pi} := \frac{\exp\left(-\frac{\kappa}{4R^2}(kX+(\gamma-k)Y)\right)}{\mathrm{tr}\left(\exp\left(-\frac{\kappa}{4R^2}(kX+(\gamma-k)Y)\right)\right)} \tag{37}$$

satisfies

$$[\pi-\tilde{\pi}]_{\mathcal{T}} \le \kappa. \tag{38}$$

Let N be the integer which minimizes $|\kappa N' - \|y^*\|_1|$ over $N' \in [\lceil\frac{\alpha}{\kappa}\rceil]$. By the bound $\|y^*\| \le \sum_i b_i y_i^* \le \alpha$, we have

$$|\kappa N - \|y^*\|_1| \le \kappa. \tag{39}$$

Then

$$\begin{aligned}\sum_{j=1}^m \overline{q}_{\rho,k_{\mathrm{opt}}}\mathrm{tr}(A_j\rho) &\ge \sum_{j=1}^m \frac{y_j^*}{\|y^*\|_1}\mathrm{tr}(A_j\rho) - (\kappa+\nu)\\ &\ge \frac{1}{\|y^*\|_1}\mathrm{tr}(C\rho) - (\kappa+\nu)\\ &\ge \frac{1}{\kappa(N+1)}\mathrm{tr}(C\rho) - (\kappa+\nu)\end{aligned} \tag{40}$$

where the first inequality follows from Eq. (38) and the fact that $\overline{q}_{\rho,k}$ is ν-close to $q_{\rho,k}$, the second from Eq. (18), and the last from Eq. (39).

Likewise,

$$\begin{aligned}\sum_{j=1}^{m} \overline{q}_{\rho,k_{\text{opt}}} b_i &\le \sum_{j=1}^{m} \frac{y_i^*}{\|y^*\|_1} b_i + R(\kappa+\nu) \\ &\le \frac{1}{\|y^*\|_1}\alpha + R(\kappa+\nu) \\ &\le \frac{1}{\kappa(N-1)}\alpha + R(\kappa+\nu). \quad (41)\end{aligned}$$

■

IV. Implementing the Oracle for $\overline{h}(\rho,\lambda,\mu)$

In this section we explain how to implement the oracle that outputs the entries of the Hamiltonian $\overline{h}(\rho,\lambda,\mu)$ defined in Eq. (6). We start with the following standard result in quantum algorithms:

Lemma 12. *Given an s-sparse $n\times n$ Hermitian matrix A with $\|A\|\le 1$ and a density matrix ρ, with probability greater than $1-p_e$, one can compute $\operatorname{tr}(\rho A)$ with additive error ε in time $O(s\varepsilon^{-2}\log^4(ns/(p_e\varepsilon)))$ using $O(\log(1/p_e)\varepsilon^{-2})$ copies of ρ.*

Proof: Using the Hamiltonian simulation technique of Ref. [25], the phase estimation algorithm takes time $(s\log^4(ns/\varepsilon)))$ to measure to accuracy $\varepsilon/2$ the energy of A_i in the state ρ. By the Chernoff bound, repeating the process $O(1/\varepsilon^2)$ times allows us to obtain an estimation for $\operatorname{tr}(\rho A_i)$ to accuracy ε. ■

Lemma 13. *Using $O(\log(m/p_e)h_{\textit{precision}}^{-2})$ copies of ρ and time $O(sh_{\textit{precision}}^{-2}\log^4(nms/(p_e h_{\textit{precision}})))$, one can implement $O[\overline{h}]$ with error probability p_e.*

Proof: Since by assumption we have an oracle for the $\{b_i\}$, we can focus on showing how to compute $\operatorname{tr}(A_i\rho)$ to accuracy ν given access to an oracle for the entries of the A_i and copies of the state ρ. Lemma 12 shows how to compute, with probability at least $1-p'_e$, an estimate of $\operatorname{tr}(A_i\rho)$ to accuracy $h_{\text{precision}}$ in time $O(sh_{\text{precision}}^{-2}\log^4(ns/(p'_e h_{\text{precision}}))$ using $O(\log(1/p'_e)h_{\text{precision}}^{-2})$ copies of ρ. Suppose the input of the oracle is $\sum_{i=1}^m c_i\,|i\rangle$. Then its output is $\sum_{i=1}^m c_i\,|i,e_i\rangle$ with e_i the estimation of $\operatorname{tr}(A_i\rho)$. We know each e_i is within $h_{\text{precision}}$ from the true value with probability at least $1-p'_e$. Therefore by the union bound all of the e_i are $h_{\text{precision}}$-close with probability $1-mp'_e$. ■

V. The Quantum Algorithm: Correctness and Complexity

We are ready to prove:

Theorem 14 (restatement of Theorem 4)**.** *Algorithm 3 runs in time*

$$\tilde{O}\left(\frac{R^{21}}{\delta^{11}}G_{\overline{h}}G_M\right)+\tilde{O}\left(\frac{R^{13}}{\delta^5}T_{\textit{Meas}}\right). \qquad (42)$$

The algorithm fails with probability at most

$$O((R/\delta)^{18}(nm)^{10}\exp(-\log^{\xi}(nm))). \qquad (43)$$

Assuming it does not fail, if it outputs Greater*, the optimal objective value is greater than $(1-\delta)\alpha$. Otherwise it outputs a sample of a probability distribution p and a real number L such that $y=Lp$ is dual feasible and $\sum_i y_i b_i \le (1+\delta)\alpha$.*

Proof:

Correctness of Algorithm: We let $p_e = \exp(-\log^{\xi}(nm))$, with p_e the error probability for the oracle $\overline{h}$. If at some call to $\overline{h}$, the output is wrong, we declare the algorithm failed. Let us first assume that the oracle $\overline{h}$ outputs the correct value in all calls. Later we will show the oracle to $\overline{h}$ is used at most $O((R/\delta)^{10}(nm)^{10}\exp(-\log^{\xi}(nm)))$ times, the algorithm fails due to a faulty $\overline{h}$ with probability at most $O((R/\delta)^{18}(nm)^{10}\exp(-\log^{\xi}(nm)))$.

Let us first consider the case in which for every $t\le T$, after step 2, $y^{(t)}$ is not equal to $(0,\dots,0)$. Then the algorithm has to output a sample from $y/\|y\|_1$ and the value of $\|y\|_1$, with

$$\overline{y}=\frac{\delta\alpha}{2R}e_1+\frac{1}{T}\sum_{t=1}^{T}y^{(t)}. \qquad (44)$$

Note $\|y^{(t)}\|_1=\varepsilon N_t$, so we know all of them. Therefore we can compute $\|\overline{y}\|_1$. Since we have samples from $y^{(t)}$ for all $t\le T$, we can sample from $\overline{y}$, which is a convex combination of them (and where we know the mixing probability distribution).

Lemma 13 with $h_{\text{precision}}=\varepsilon/2$ and Lemma 11 with $\nu=\kappa=\varepsilon/2$ show Step 2 of the algorithm implements $\mathsf{ORACLE}(\rho^{(t)})$ as in Algorithm 10. Then for each t, Step 2 outputs a vector $y^{(t)}$ such that, with probability at least $1-\exp(-\log^{\xi}(nm))$,

$$\sum_{i=1}^{m} b_i y_i^{(t)} \le \alpha(1+2R\varepsilon), \qquad (45)$$

and

$$\sum_{i=1}^{m} y_i^{(t)}\operatorname{tr}(A_i\rho^{(t)}) \ge \operatorname{tr}(C\rho^{(t)})-2\alpha\varepsilon. \qquad (46)$$

The remaining steps of the algorithm run the Matrix Multiplicative Weight method. One difference (which will be important to keep the quantum complexity of the algorithm low), is the sparsification of the pay-off matrix $M^{(t)}$. Let

$$\hat{M}^{(t)} := \left(\sum_{i=1}^{m} y_i^{(t)} A_i - C + 2\alpha I \right) / 4\alpha. \quad (47)$$

Since

$$\left\| \sum_{i=1}^{m} y_i^{(t)} A_i - C \right\| \leq \sum_{i=1}^{m} y_i^{(t)} + 1 \leq \sum_{i=1}^{m} b_i y_i^{(t)} + 1 \leq \alpha + 1, \quad (48)$$

we have $0 \leq \hat{M}^{(t)} \leq I$.

By the Matrix Hoeffding bound (Lemma 15), with probability greater than $1 - \exp(-\log^{\xi}(nm))$,

$$\left\| \hat{M}^{(t)} - M^{(t)} \right\| \leq \frac{1}{4T}. \quad (49)$$

Then by Lemma 16 and the fact the Gibbs sampler has error $\varepsilon/2$, we find

$$\left\| \rho^{(t)} - \hat{\rho}^{(t)} \right\|_1 \leq \varepsilon, \quad (50)$$

with

$$\hat{\rho}^{(t)} := \frac{\exp(-\varepsilon'((\sum_{\tau=1}^{t} \hat{M}^{(\tau)}))}{\mathrm{tr}\left(\exp(-\varepsilon'((\sum_{\tau=1}^{t} \hat{M}^{(\tau)}))\right)}. \quad (51)$$

We are ready to show the correctness of the algorithm. Since $\eta = \frac{\delta\alpha}{2R}$, we find

$$\begin{aligned} \overline{y}.b &= R\eta + \frac{1}{T}\sum_{t=1}^{T} y^{(t)}.b \\ &\leq R\eta + \alpha(1 + 2R\varepsilon) \\ &= \alpha\left(1 + \delta/2 + 2R\varepsilon\right) \\ &\leq (1+\delta)\alpha. \quad (52) \end{aligned}$$

Let us now show that $\overline{y}$ is dual feasible. It is clear that $\overline{y} \geq 0$. In analogy with the proof of Theorem 8 (Theorem 1 of [9]), we have

$$\begin{aligned} & \lambda_n\left(\sum_{j=1}^{m} \overline{y}_j A_j - C\right) \quad (53) \\ = \; & \lambda_n\left(\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{m} y_j^t A_j - C\right) + \eta \quad (54) \\ = \; & 4\alpha\lambda_n\left(\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{j=1}^{m} y_j^t A_j - C + 2\alpha I\right)/4\alpha\right) \\ - \; & 2\alpha + \eta \quad (55) \\ \overset{(i)}{\geq} \; & \frac{4\alpha}{(1+\varepsilon)}\frac{1}{T}\sum_{t=1}^{T}\mathrm{tr}\left(\hat{\rho}^{(t)}\left(\sum_{j=1}^{m} y_j^t A_j - C + 2\alpha I\right)/4\alpha\right) \\ - \; & \frac{4\alpha\ln(n)}{T(1+\varepsilon)\varepsilon} - 2\alpha + \eta \quad (56) \\ \overset{(ii)}{\geq} \; & \frac{4\alpha}{(1+\varepsilon)}\frac{1}{T}\sum_{t=1}^{T}\mathrm{tr}\left(\rho^{(t)}\left(\sum_{j=1}^{m} y_j^t A_j - C + 2\alpha I\right)/4\alpha\right) \\ - \; & \frac{4\alpha\varepsilon}{(1+\varepsilon)} - \frac{4\alpha\ln(n)}{T(1+\varepsilon)\varepsilon} - 2\alpha + \eta \quad (57) \\ \overset{(iii)}{\geq} \; & \frac{4\alpha}{(1+\varepsilon)}\left(\frac{1}{2} - 2\alpha\varepsilon\right) - \frac{4\alpha\varepsilon}{(1+\varepsilon)} - \frac{4\alpha\ln(n)}{T(1+\varepsilon)\varepsilon} \\ - \; & 2\alpha + \eta \quad (58) \\ = \; & -\frac{6\alpha\varepsilon}{(1+\varepsilon)} - \frac{8\alpha^2\varepsilon}{(1+\varepsilon)} + \frac{\alpha\delta}{2R} - \frac{4\alpha\ln(n)}{T(1+\varepsilon)\varepsilon} \\ \geq \; & \frac{\delta\alpha}{4R} - \frac{4\alpha\ln(n)}{T(1+\varepsilon)\varepsilon} \quad (59) \\ \geq \; & 0, \quad (60) \end{aligned}$$

where we used that $\alpha \geq 1$,

$$T = \frac{16R\ln(n)}{\delta\varepsilon}, \quad (61)$$

and

$$\varepsilon = \frac{\delta}{28R^2}. \quad (62)$$

Inequality (i) follows from the Matrix Multiplicative Weight method (Lemma 7), which can be applied since by Eq. (48),

$$0 \leq \left(\sum_{j=1}^{m} y_j^t A_j - C + 2\alpha I\right)/4\alpha \leq I. \quad (63)$$

Inequality (ii) follows from Eq. (50), while Inequality (iii) follows from Eq. (46).

Let us now turn to the case in which there is a $t \in [T]$ such that the loop in Step 2 outputs $y^{(t)} = (0, \ldots, 0)$.

Then by the Chernoff bound and Lemma 9 we find that for every $y \geq 0$ such that

$$\sum_{i=1}^{m} y_i \operatorname{tr}(A_i\rho) \geq \operatorname{tr}(C\rho) - 3\alpha\varepsilon, \tag{64}$$

we must have

$$y.b \leq \alpha(1 - 3R\varepsilon). \tag{65}$$

By duality of linear programming it follows the maximum over $\lambda \geq 0$ of

$$\lambda(\operatorname{tr}(C\rho^{(t)}) - 3\alpha\varepsilon) \tag{66}$$

subject to the constraints

$$\lambda \operatorname{tr}(A_i\rho^{(t)}) \leq b_i, \;\; i \in [m] \tag{67}$$

must be greater than $\alpha(1 - 3R\varepsilon)$. Note that $\lambda = \lambda \operatorname{tr}(\rho^{(t)}) \leq R$. Then defining $X = \lambda_{\text{optimal}}\rho^{(t)}$ (with λ_{optimal} the optimal value of λ for the LP above), we find that $\operatorname{tr}(A_iX) \leq b_i$ for all $i \in [m]$ and

$$\operatorname{tr}(CX) \geq \alpha(1 - 3R\varepsilon) \geq (1 - \delta)\alpha. \tag{68}$$

Finally, let us bound the probability that the algorithm fails. This can be due to three causes. The first is a faulty oracle call for $\overline{h}$. This is upper bounded by

$$O\left((R/\delta)^{18}(nm)^{10}\exp\left(-\log^{\xi}(nm)\right)\right). \tag{69}$$

The second is due to an error in estimating the values of $\{\operatorname{tr}(A_i\rho^{(t)})\}$ in Step 2 of the algorithm. The third is the random sampling used to construct $M^{(t)}$. By the union bound the error probability of both are also bounded by Eq. (69).

Run-time Analysis: We remind the reader we are assuming that $\alpha \geq 1$.

The cost of running step 2 is $\tilde{O}(\alpha R^2/\varepsilon^3) \leq \tilde{O}(R^9/\delta^3)$ times the sum of cost of the following: (i) performing M times the Gibbs sampling of the Hamiltonian $\overline{h}$ with cost $MG_{\overline{h}}$, (ii) sampling M times from the resulting distribution with cost $O(M)$, and (iii) computing estimates to the expectation values of the input matrices on the state $\rho^{(t)}$, with cost $(M+1)T_{\text{Meas}}$. Therefore the total cost of step 2 (for a particular $t \in [T]$) is

$$\tilde{O}\left(\frac{R^9}{\delta^3}M\left(G_{\overline{h}} + T_{\text{Meas}}\right)\right) = \tilde{O}\left(\frac{R^{13}}{\delta^5}\left(G_{\overline{h}} + T_{\text{Meas}}\right)\right). \tag{70}$$

The cost of step 4 is $(Q+1)G_{\overline{h}} = \tilde{O}((R^6/\delta^4)G_{\overline{h}}$. The cost of step 5 is $Q = \tilde{O}(R^6/\delta^4)$. The cost of steps 6-7 is

$$\begin{aligned} C_tG_M = \tilde{O}\left(\frac{\alpha}{\varepsilon^3}\frac{R^2}{\varepsilon^2}\left(\frac{1}{\varepsilon^2} + \frac{R^6}{\delta^4}\right)G_{\overline{h}}\right)G_M \\ + \tilde{O}\left(\frac{2\alpha}{\varepsilon^3}\frac{R^2}{\varepsilon^4}\right)G_M = \left(\frac{R^{19}}{\delta^9}\right)G_{\overline{h}}G_M. \end{aligned} \tag{71}$$

As each step is repeated $T = \tilde{O}(R^3/\delta^2)$ times, the total cost is

$$\tilde{O}\left(\frac{R^{21}}{\delta^{11}}G_{\overline{h}}G_M\right) + \tilde{O}\left(\frac{R^{13}}{\delta^5}T_{\text{Meas}}\right). \tag{72}$$

■

Lemma 15. *(Matrix Hoeffding Bound, Theorem 2.8 of [26]) Suppose $Z_1, \ldots, Z_k$ are independent random $d \times d$ Hermitian matrices satisfying $\mathbb{E}[Z_i] = 0$ and $\|Z_i\| \leq \lambda$. Then*

$$Pr\left[\left\|\frac{1}{k}\sum_{i=1}^{k} Z_i\right\| \geq \delta\right] \leq d.e^{-\frac{k\delta^2}{8\lambda^2}}. \tag{73}$$

Lemma 16. *Let H, H' be Hermitian matrices. Then*

$$\left\|\frac{e^H}{\operatorname{tr}(e^H)} - \frac{e^{H'}}{\operatorname{tr}(e^{H'})}\right\|_1 \leq 2\left(e^{\|H-H'\|} - 1\right). \tag{74}$$

The proof of the lemma above is given in the full version of this paper.

Finally let us prove

Corollary 17 (Restatement Corollary 5)**.** *Using the Gibbs Sampler from Ref. [13], Algorithm 3 runs in time $\tilde{O}(n^{\frac{1}{2}}m^{\frac{1}{2}}s^2R^{32}/\delta^{18})$.*

The proof is given in the full version of the paper.

Acknowledgments

We thank Joran van Apeldoorn, Ronald de Wolf, Andras Gilyen, Aram Harrow, Sander Gribling, Matt Hastings, Cedric Yen-Yu Lin, Ojas Parekh, and David Poulin for interesting discussions and useful comments on the paper. This work was funded by Cambridge Quantum Computing, Microsoft and the National Science Foundation.

References

[1] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41.2, 303 (1999).

[2] L.K. Grover. Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. 79, 325 (1997).

[3] L. Vandenberghe and S. Boyd. Semidefinite programming. SIAM Review 38, 49 (1996).

[4] M.X. Goemans. Semidefinite programming in combinatorial optimization. Mathematical Programming 79, 143 (1997).

[5] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge University Press, 2004.

[6] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42, 1115 (1995).

[7] Y.T. Lee, A. Sidford, and S.C. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. IEEE 56th Annual Symposium on the Foundations of Computer Science (FOCS), 2015.

[8] S. Arora, E. Hazan and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. 46th Annual IEEE Symposium on Foundations of Computer Science, 2005. FOCS 2005.

[9] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. ACM, 2007.

[10] S. Arora, E. Hazan and S. Kale. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. Theory of Computing 8, 121 (2012).

[11] K. Temme et al. Quantum metropolis sampling. Nature 471, 87 (2011).

[12] M.H. Yung and A. Aspuru-Guzik. A quantumquantum Metropolis algorithm. Proceedings of the National Academy of Sciences 109, 754 (2012).

[13] D. Poulin and P. Wocjan. Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer. Phys. Rev. Lett. 103, 220502 (2009).

[14] A.N. Chowdhury and R.D. Somma. Quantum algorithms for Gibbs sampling and hitting-time estimation. arXiv preprint arXiv:1603.02940 (2016).

[15] M. Kastoryano and F.G.S.L. Brandao. Quantum Gibbs Samplers: the commuting case. Comm. Math. Phys. 344, 915 (2016).

[16] F.G.S.L. Brandao and M. Kastoryano. Finite correlation length implies efficient preparation of quantum thermal states. In preparation.

[17] G. Brassard et al. Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53 (2002).

[18] E.T. Jaynes. Information Theory and Statistical Mechanics II. Phys. Rev. 108, 171 (1957).

[19] J.R. Lee, P. Raghavendra, and D. Steurer. Lower bounds on the size of semidefinite programming relaxations. Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing. ACM, 2015.

[20] R. Ahlswede, A. Winter. Strong Converse for Identification via Quantum Channels". IEEE Trans. Information Theory 48, 569 (2003).

[21] N.E. Sherman, T. Devakul, M.B. Hastings, R.R.P. Singh. Phys. Rev. E 93, 022128 (2016).

[22] S. Lloyd, M. Mohseni, P. Rebentrost. Quantum Principal Component Analysis. Nature Physics 10, 631 (2014).

[23] A. Childs and R. Kothari. Limitations on the simulation of non-sparse Hamiltonians. arXiv preprint arXiv:0908.4398 (2009).

[24] W.R. Gilks. Markov chain monte carlo. John Wiley and Sons, Ltd. Chicago (2005).

[25] D.W. Berry, A.M. Childs, R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS 2015), 792 (2015).

[26] J. A. Tropp. User-friendly tail bounds for sums of random matrices, 2010, arXiv:1004.4389.